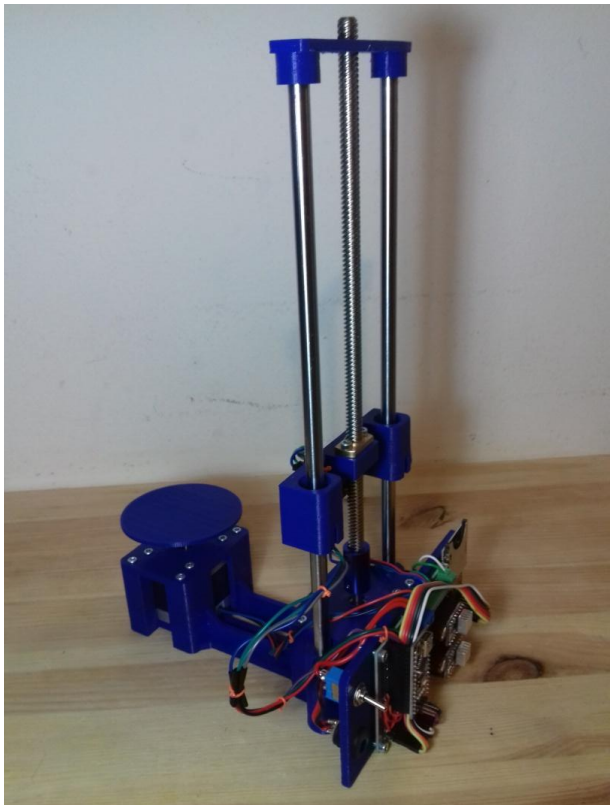
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΜΕΛΕΤΗ, ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ ΚΑΙ ΚΑΤΑΣΚΕΥΗ ΤΡΙΣΔΙΑΣΤΑΤΟΥ ΣΑΡΩΤΗ ΑΝΤΙΚΕΙΜΕΝΩΝ



ΣΠΟΥΔΑΣΤΗΣ: ΜΕΝΟΥΝΟΣ ΠΑΝΑΓΙΩΤΗΣ (Α.Μ.: 7041)

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΚΑΛΑΡΑΚΗΣ ΑΛΕΞΑΝΔΡΟΣ ΛΕΚΤΟΡΑΣ

ΠΑΤΡΑ 2020

ΠΡΟΛΟΓΟΣ

Το παρών τεύχος αποτελεί την πτυχιακή εργασία που εκπονήθηκε στα πλαίσια του προγράμματος σπουδών του τμήματος της Μηχανολογίας στο Τ.Ε.Ι. Δυτικής Ελλάδας και αναφέρεται στην αναλυτική μελέτη, σχεδίαση, κατασκευή και προγραμματισμό ενός τρισδιάστατου σαρωτή.

Στόχος της εργασίας είναι η κατασκευή ενός τρισδιάστατου σαρωτή χαμηλού κόστους και αυτοματοποιημένης διαδικασίας σάρωσης. Σε αρχικό στάδιο γίνεται η ψηφιακή σχεδίαση της διάταξης και του ηλεκτρονικού κυκλώματος και στην συνέχεια η υλοποίηση της κατασκευής καθώς και η δημιουργία του λογισμικού ελέγχου και λειτουργίας.

Θέλω να ευχαριστήσω πρώτα από όλα θερμά τον επιβλέποντα καθηγητή Αλέξανδρο Καλαράκη για την πολύτιμη καθοδήγηση και εμπιστοσύνη που μου έδειξε καθώς επίσης τον κ. Σωτήριο Τσίρκα για την συμβολή των ιδεών του καθώς και την κ. Γεωργία Τσώλου για την βοήθεια της στα θέματα σχετικά με τον προγραμματισμό.

Το μεγαλύτερο ευχαριστώ το οφείλω στους γονείς μου Γεώργιο και Ελένη για την οικονομική και ψυχολογική στήριξη που μου παρείχαν παρά τις δυσκολίες τόσα χρόνια δίνοντας μου την ευκαιρία να ακολουθήσω αυτό που μου αρέσει.

Υπεύθυνη Δήλωση Φοιτητή: Ο κάτωθι υπογεγραμμένος Φοιτητής έχω επίγνωση των συνεπειών του Νόμου περί λογοκλοπής και δηλώνω υπεύθυνα ότι είμαι συγγραφέας αυτής της Διπλωματικής Εργασίας, έχω δε αναφέρει στην Βιβλιογραφία μου όλες τις πηγές τις οποίες χρησιμοποίησα και έλαβα ιδέες ή δεδομένα. Δηλώνω επίσης ότι, οποιοδήποτε στοιχείο ή κείμενο το οποίο έχω ενσωματώσει στην εργασία μου προερχόμενο από Βιβλία ή άλλες εργασίες ή το διαδίκτυο, γραμμένο ακριβώς ή παραφρασμένο, το έχω πλήρως αναγνωρίσει ως πνευματικό έργο άλλου συγγραφέα και έχω αναφέρει ανελλιπώς το όνομά του και την πηγή προέλευσης.

Ο Φοιτητής
(Ονοματεπώνυμο)

……
(Υπογραφή)

ΠΕΡΙΛΗΨΗ

Η τεχνολογία της τρισδιάστατης σάρωσης είναι μια διαδικασία ανάλυσης, αντιγραφής και μετατροπής αντικειμένων ή χώρων του πραγματικού κόσμου σε ψηφιακά μοντέλα που περιέχουν γεωμετρικές και χρωματικές πληροφορίες. Ένας τρισδιάστατος σαρωτής μπορεί να χρησιμοποιεί διάφορες υπάρχοντες τεχνολογίες, η καθεμία με τους δικούς της περιορισμούς, πλεονεκτήματα και κόστος. Σημαντικό ρόλο παίζει η διαφάνεια και η ανακλαστικότητα των αντικειμένων που μπορούν να ψηφιοποιηθούν με τις αντίστοιχες κατηγορίες. Η τεχνολογία αυτή βρίσκει εφαρμογή στην παραγωγή ταινιών και βιντεοπαιχνιδιών, για την καταγραφή και αναγνώριση κινήσεων, αντίστροφη μηχανική, επιθεώρηση και ποιοτικό έλεγχο προϊόντων καθώς και την ιατρική.

Αντικείμενο αυτής της πτυχιακής εργασίας είναι η μελέτη, ο σχεδιασμός, η κατασκευή και ο προγραμματισμός ενός τρισδιάστατου σαρωτή. Ο σαρωτής διαθέτει έναν αισθητήρα απόστασης που κινείται στον κάθετο άξονα και μία περιστρεφόμενη τράπεζα, έτσι με αυτό τον τρόπο επιτυγχάνεται η δειγματοληπτική μέτρηση σημείων συντεταγμένων της επιφάνειας ενός αντικειμένου. Ένας μικροελεγκτής Arduino Nano κατάλληλα προγραμματισμένος ελέγχει την διαδικασία σάρωσης και αποθηκεύει τις μετρήσεις σε μία κάρτα μνήμης ή τις στέλνει σε πραγματικό χρόνο μέσω καλωδίου USB στον υπολογιστή. Το λογισμικό που έχει υλοποιηθεί σε Python αναλαμβάνει να εξάγει το νέφος σημείων της επιφάνειας από τις αποθηκευμένες μετρήσεις και στην συνέχεια ένα πρόγραμμα σε MATLAB ενσωματωμένο στην εφαρμογή επιχειρεί να ανακατασκευάσει την επιφάνεια του αντικειμένου.

Στο πρώτο κεφάλαιο γίνονται αναφορές σε εισαγωγικές έννοιες όπως η αντίστροφη μηχανική και τεχνολογίες σχεδίασης ανάλυσης και ελέγχου παραγωγής, καθώς και διαφόρων υπαρχόντων τεχνολογιών τρισδιάστατης σάρωσης.

Στο δεύτερο κεφάλαιο περιγράφονται διάφοροι αλγόριθμοι ανακατασκευής επιφανειών με δεδομένο το νέφος σημείων της επιφάνειας.

Στο τρίτο κεφάλαιο γίνεται ανάλυση της αρχής λειτουργίας του σαρωτή, του ελέγχου των βηματικών κινητήρων και των πρωτοκόλλων επικοινωνίας τα οποία χρησιμοποιούνται μεταξύ του κεντρικού μικροελεγκτή της διάταξης και των περιφερειακών.

Στο τέταρτο κεφάλαιο παρέχονται πληροφορίες σχετικά με διάφορους μικροελεγκτές και αισθητήρες του εμπορίου και γίνεται η αρχική σχεδίαση καθώς και η υλοποίηση του ηλεκτρονικού κυκλώματος της διάταξης σάρωσης.

Στο πέμπτο κεφάλαιο γίνεται ψηφιακός σχεδιασμός των μηχανικών μερών της κατασκευής στο σχεδιαστικό πρόγραμμα *Solidworks* και τα επιμέρους κομμάτια τυπώνονται σε τρισδιάστατο εκτυπωτή. Η διάταξη τελικά συναρμολογείται με τα ηλεκτρονικά στοιχεία.

Στο έκτο κεφάλαιο υλοποιείται το λογισμικό του σαρωτή το οποίο αποτελείται από το μέρος που εκτελείται στον μικροελεγκτή, και την εφαρμογή στην πλευρά του υπολογιστή χτισμένο σε Python μαζί με ενσωματωμένο έναν κώδικα σε MATLAB για την ανοικοδόμηση της επιφάνειας. Ο πηγαίος κώδικα μεταγλωττίζεται και δημιουργείται και το εκτελέσιμο αρχείο εγκατάστασης της εφαρμογής στο Inno Setup.

Στο έβδομο κεφάλαιο εκτελείται μία ολοκληρωμένη διαδικασία σάρωσης, από το φυσικό αντικείμενο μέχρι το ψηφιακό μοντέλο.

Τέλος αναφέρονται τα συμπεράσματα και οι προτάσεις βελτίωσης της διάταξης καθώς παρέχονται τα μηχανολογικά σχέδια, το σχηματικό διάγραμμα κυκλώματος καθώς και ο πηγαίος κώδικας.

ΠΡΟΛΟΓΟΣ ..... **iii**
ΠΕΡΙΛΗΨΗ ..... **iv**
ΕΙΣΑΓΩΓΗ **1**

1. Επιστήμη της μηχανικής 1
2. Αντίστροφη μηχανική 1
3. Συστήματα CAD/CAE/CAM 3
4. Τεχνολογίες 3D σάρωσης 5
5. Επιλογή είδους σάρωσης 12

2.ΑΛΓΟΡΙΘΜΟΙ ΑΝΑΚΑΤΑΣΚΕΥΗΣ ΕΠΙΦΑΝΕΙΩΝ **13**

2.1 Αλγόριθμος Jarvis 14
2.2 Αλγόριθμος Quickhull 14
2.3 Αλγόριθμος Graham 15
2.4 Incremental αλγόριθμος 17
2.5 Διαιρεί και βασίλευε 18
2.6 Τριγωνισμός Delaunay 22
2.7 Boissonnat 23
2.8 Αλγόριθμος μηδενικού συνόλου Hoppe 24
2.9 Αλγόριθμος Crust 26

3.ΑΡΧΗ ΛΕΙΤΟΥΡΓΙΑΣ **28**

3.1 Λειτουργία σαρωτή 28
3.2 Λειτουργία βηματικών κινητήρων 30
3.3 UART 38
3.4 SPI 40
3.5 I2C 43

4.ΗΛΕΚΤΡΟΝΙΚΟ ΚΥΚΛΩΜΑ **46**

4.1 Γενικά 46
4.2 Μικροεπεξεργαστές και μικροελεγκτές 50
4.3 Επιλογή ελεγκτή 51
4.4 Οδηγός κινητήρα 53
4.5 Αισθητήρας TOF 54
4.6 Σχεδιασμός πλακέτας 55
4.7 Υλοποίηση πλακέτας 57

5.ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΚΑΤΑΣΚΕΥΗ ΣΑΡΩΤΗ **61**

5.1 Ψηφιακή σχεδίαση 61
5.2 Έλεγχος συναρμογής 65
5.3 3D Εκτύπωση των εξαρτημάτων 66
5.4 Τελική συναρμολόγηση σαρωτή 67

6. ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ **69**

6.1 Κώδικας Arduino 69
6.2 Κώδικας Python 70
6.3 Κώδικας MATLAB 72
6.3 Εκτελέσιμο αρχείο 73

7. ΔΙΑΔΙΚΑΣΙΑ ΣΑΡΩΣΗΣ **75**

7.1 Σάρωση με κάρτα μνήμης 75
7.2 Σάρωση μέσω USB 76

7.3 Παραγωγή νέφους σημείων 76
7.4 Λογισμικό MeshLab 76
7.6 Δημιουργία STL αρχείου 79
8. ΣΥΜΠΕΡΑΣΜΑΤΑ 80
8.1 Συνολικό κόστος 80
8.2 Βελτιώσεις 82
ΒΙΒΛΙΟΓΡΑΦΙΑ 83
ΠΑΡΑΡΤΗΜΑ Α 90
ΠΑΡΑΡΤΗΜΑ Β 91
ΠΑΡΑΡΤΗΜΑ Γ 92

ΕΙΣΑΓΩΓΗ

1. Επιστήμη της μηχανικής

Ο ευρύς τομέας της μηχανικής μπορεί να οριστεί ως η διαδικασία επίλυσης προβλημάτων και αναγκών αξιοποιώντας τις αρχές των μαθηματικών και της φυσικής [δ1]. Περιλαμβάνει μία σειρά διαδικασιών όπως η έρευνα, μελέτη, σχεδίαση, ανάπτυξη, εφαρμογή, κατασκευή, συντήρηση και βελτίωση συστημάτων, δομών, εγκαταστάσεων και μηχανών. Η επιστήμη της μηχανικής απαρτίζεται από διάφορους ακαδημαϊκούς και επαγγελματικούς κλάδους με κύριους:

- Χημικής
- Αρχιτεκτονικής
- Πολιτικής
- Ηλεκτρολογίας
- Μηχανολογίας
- Περιβάλλοντος
- Υπολογιστών

2. Αντίστροφη μηχανική

Ορισμός

Πρόκειται για τις διαδικασίες αποδόμησης ενός τελικού προϊόντος ώστε να φανερωθούν οι αρχικές ιδιότητες με σκοπό την αντιγραφή ή την βελτίωση αυτού χωρίς να υπάρχει πρόσβαση στις αρχικές πληροφορίες σχεδίασης. Είναι ένα πολύ συχνό φαινόμενο στην αγορά καθώς πολλές εταιρείες μελετούν και αναλύουν προϊόντα από τους ανταγωνιστές τους με σκοπό να συγκρίνουν και να βελτιώσουν τα δικά τους ή ακόμα και να παράγουν πανομοιότυπα. Άλλοι λόγοι που μπορεί να εφαρμοστεί η αντίστροφη μηχανική είναι για να επανασχεδιαστούν αρχαίου τύπου προϊόντα σε ψηφιακή μορφή η ακόμα και για εκπαιδευτικούς λόγους. Η αντίστροφη μηχανική βρίσκει εφαρμογές σε κλάδους όπως της:

- Χημικής μηχανικής
- Λογισμικού υπολογιστών
- Μηχανολογίας
- Ηλεκτρονικών συστημάτων.

Χημική μηχανική

Στον τομέα της χημικής μηχανικής γίνεται προσπάθεια ανάλυσης και προσδιορισμού των χημικών στοιχείων η ουσιών που είναι ενσωματωμένα στο προϊόν καθώς και πώς αυτά αλληλεπιδρούν [δ2]. Τα αποτελέσματα των χημικών αναλύσεων μπορεί να χρησιμοποιηθούν για την αντιγραφή ή τον ποιοτικό έλεγχο του προϊόντος. Μία ανάλυση πρώτης τάξης ενός χημικού μείγματος μπορεί να περιλαμβάνει τον διαχωρισμό των φάσεων. Μέθοδοι που χρησιμοποιούνται είναι:

- Φυγοκέντριση
- Εκχύλιση

- Διήθηση.

Μία ανάλυση δεύτερης τάξης περιλαμβάνει την χημική ανάλυση των επιμέρους φάσεων. Για τα υγρά χρησιμοποιούνται κυρίως ο:

- Χρωματογραφικός διαχωρισμός
- Απόσταξη.

Στα στερεά αντίστοιχα μερικοί από τις μεθόδους που χρησιμοποιούνται είναι:

- Πυρόλυση
- Θερμιδομετρία
- Μικροσκοπία αντίστοιχα.

Πληροφορική

Στον κλάδο της πληροφορικής και του προγραμματισμού η δημιουργία λογισμικού ξεκινάει με την:

1. Σύνταξη του πηγαίου κώδικα
2. Συντακτική διόρθωση μέσω αποσφαλματωτή
3. Λογική αποσφάλματωση
4. Παραγωγή του αντικειμενικού κώδικα σε δυαδική μορφή μέσω του μεταγλωττιστή.

Οι διαδικασίες που αποσκοπούν στην ανασύνταξη του πηγαίου κώδικα η την παράκαμψη δικλείδων ασφαλείας του λογισμικού ονομάζεται αντίστροφη μηχανική λογισμικού [δ3]. Η αντιστροφή του δυαδικού κώδικα στον πηγαίο μπορεί να έχει στόχο την αντιγραφή ή τον έλεγχο του βαθμού ασφαλείας του λογισμικού.

Ηλεκτρονικά συστήματα

Τα ηλεκτρονικά συστήματα μπορούν και αυτά να αποσυναρμολογηθούν και να αναλυθούν για να ανακτηθούν έτσι τα αρχικά χαρακτηριστικά σχεδίασης [δ4]. Η διαδικασία αυτή μπορεί να έχει σκοπό την αντιγραφή λόγω ανταγωνισμού, την ανάκτηση των δεδομένων παραγωγής σε προϊόντα που πλέον δεν παράγονται ή ακόμα και τα όρια αντοχής.

Μηχανολογία

Η αντίστροφη μηχανική στον τομέα της μηχανολογίας περιλαμβάνει τις διαδικασίες που ακολουθούνται για την εύρεση φυσικών διαστάσεων και χαρακτηριστικών που διέπουν ένα αντικείμενο. Στην πιο απλή μορφή της η εξαγωγή των πληροφοριών μπορεί να γίνει με μέτρηση των διαστάσεων των ακμών των αντικειμένων με χρήση μετρητικών οργάνων απόστασης όπως για παράδειγμα ο χάρακας και το παχύμετρο.

Με την εξέλιξη των υπολογιστών και των σχεδιαστικών προγραμμάτων CAD τα φυσικά αντικείμενα μπορούν πλέον να αναπαρασταθούν σε ψηφιακή μορφή. Για να μπορέσει να δημιουργηθεί ένα ψηφιακό αντίγραφο ενός ήδη υπάρχοντος προϊόντος γίνεται χρήση των τρισδιάστατων σαρωτών που σκοπός τους είναι να

δημιουργηθεί σε ψηφιακή μορφή η φυσική γεωμετρία ενός αντικειμένου. Συνήθως χρησιμοποιείται για την αντιγραφή και την παραγωγή διπλότυπων. Παρόλα αυτά μπορεί να χρησιμοποιηθεί για την ανάκτηση των χαρακτηριστικών παλαιότερων προϊόντων που έχει σταματήσει η παραγωγή τους, για την δημιουργία ψηφιακού αρχείου του αντικειμένου, αξιολόγηση, αλλά και για εκπαιδευτικούς σκοπούς.

3. Συστήματα CAD/CAE/CAM

Συστήματα CAD

Αρχικά όλα τα κατασκευαστικά σχέδια γίνονταν στο χέρι με μολύβι και χαρτί από ειδικευμένους σχεδιαστές κάνοντας χρήση συμβατικών σχεδιαστικών εργαλείων ακολουθώντας μία σειρά κανόνων. Αυτό καθιστούσε την διαδικασία σχεδίασης χρονοβόρα και κουραστική. Η σχεδίαση με χρήση ηλεκτρονικού υπολογιστή (*Computer Aided Design*) δίνει την δυνατότητα τα τεχνικά και κατασκευαστικά σχέδια να αναπαρίστανται σε ψηφιακή μορφή κάνοντας εύκολη την επεξεργασία και την μεταφορά αυτών, ο χρόνος επεξεργασίας και δημιουργίας έχει μειωθεί δραματικά. Τα σχέδια μπορούν να προβληθούν σε πραγματικό χρόνο από διάφορες οπτικές γωνίες διασφαλίζοντας έτσι την σωστή εφαρμογή και τον σχεδιασμό του τελικού αποτελέσματος.

Τέσσερις είναι οι βασικές αρχές αναπαράστασης που εφαρμόζονται κατά την σχεδίαση με χρήση συστημάτων *CAD* [δ5]:

Συρματική αναπαράσταση

Πρόκειται για την απλούστερη μορφή αναπαράστασης ενός αντικειμένου χωρίς να δίνονται πληροφορίες για το σχήμα των επιφανειών ή να διαχωρίζεται το εσωτερικό από το εξωτερικό και καθορίζεται μόνο από τις ακμές του. Το αντικείμενο προβάλλεται ως ένα σύνολο από ευθείες ή καμπύλες γραμμές χωρίς μεγάλη ευκρίνεια χαρακτηριστικών προβάλλοντας την βασική γεωμετρία στο χώρο, παρόλα αυτά δεν απαιτείται μεγάλη υπολογιστική και αποθηκευτική ισχύ για την δημιουργία και επεξεργασία τους.

Επιφανειακή μοντελοποίηση

Κατά την επιφανειακή μοντελοποίηση το αντικείμενο παρουσιάζεται με στερεές επιφάνειες χωρίς πάχος προβάλλοντας έτσι ευκρινέστερα της προηγούμενης τεχνικής τα επιφανειακά χαρακτηριστικά του αντικειμένου σε διαφορετικές οπτικές γωνίες. Μέσω της επιφανειακής μοντελοποίησης μπορούν να οριστούν χαρακτηριστικά της επιφάνειας και να αποδοθούν τα αληθοφανή χαρακτηριστικά μέσω γραφικών. Παρέχουν αρκετά δεδομένα για να χρησιμοποιηθούν ως είσοδο σε μηχανές αυτοματοποιημένης κατεργασίας *CNC* αλλά και για την κατασκευή καλουπιών.

Στερεά μοντελοποίηση

Αποτελεί την πληρέστερη τεχνική ψηφιακής αναπαράστασης αντικειμένων στο χώρο. Χαρακτηριστικά των φυσικών μεγεθών όπως όγκος, βάρος, ροπή αδρανείας είναι πλέον υπολογίσιμα. Η δημιουργία ενός στερεού μπορεί να γίνει με συνδυασμό βασικών γεωμετριών όπως κύλινδροι, κύβοι, σφαίρες ή από οριοθετημένες

επιφάνειες όπου η σάρωση αυτών στο χώρο θα δώσει τον ζητούμενο όγκο του αντικειμένου

Παραμετρική μοντελοποίηση

Σε αυτή την τεχνική τα αντικείμενα σχεδιάζονται δυναμικά δίνοντας έτσι την δυνατότητα στον χρήστη να επεμβαίνει στις διαστάσεις του αντικειμένου σε μεταγενέστερο χρόνο. Κάθε διάσταση στο χώρο περιγράφεται από μία μαθηματική εξίσωση οι οποία μπορεί να συνδυάζεται με άλλες. Η αλλαγή σε μία από αυτές θα επιφέρει και αλλαγή σε όλες τις εμπλεκόμενες και το μοντέλο θα ανασχεδιαστεί αυτόματα. Αυτό έχει ως αποτέλεσμα την δημιουργία ενός ευέλικτου μοντέλου το οποίο μπορεί να επεξεργαστεί και να διορθωθεί πολύ εύκολα.

Τα λογισμικά *CAD* μπορεί να διαφέρουν ανάλογα με την ειδικότητα ενός μηχανικού. Υπάρχει μία ποικιλία από εξειδικευμένα εργαλεία σχεδίασης με πιο δημοφιλή:

- *Solidworks*
- *KATAIA*
- *Fusion 360*
- *AutoCAD*

Συστήματα CAE

Εφόσον ένα αντικείμενο σχεδιαστεί σε δεύτερη φάση είναι αναγκαίο να εκτελεστούν όλοι οι απαραίτητοι υπολογισμοί που θα καθορίσουν αν αυτό είναι λειτουργικό κάτω από την επίδραση των συνθηκών του περιβάλλοντος λειτουργίας του [δ6]. Τα συστήματα *CAE* (*Computer Aided Engineering*) επιτυγχάνουν την μελέτη, ανάλυση και προσομοίωση συστημάτων με χρήση ηλεκτρονικού υπολογιστή και εφαρμόζεται σε συστήματα όπως:

- Τάσεων παραμορφώσεων
- Μετάδοσης θερμότητας
- Ρευστομηχανικής
- Μαγνητικών πεδίων
- Ακουστικής
- Μηχανικής εδάφους

Η μαθηματική ανάλυση γίνεται μέσω της μεθόδου των πεπερασμένων στοιχείων. Στην ουσία το συνολικό μοντέλο χωρίζεται σε μικρά απλούστερα στοιχεία τμήματα αλληλοσυνδεόμενα μεταξύ τους και για κάθε στοιχείο επιλύονται αλγεβρικά ή αριθμητικά οι εξισώσεις που τα διέπουν. Στην συνέχεια συνδυάζονται οι λύσεις αποδίδοντας έτσι το αποτέλεσμα του συνολικού συστήματος. Η ροή ενεργειών που ακολουθείται από τα λογισμικά CAE περιλαμβάνει:

1. Εξαγωγή πρότυπου μοντέλου ανάλυσης πεπερασμένων (προ-επεξεργαστής)
2. Επίλυση ανάλυσης πεπερασμένων με αριθμητικές μεθόδους (επεξεργαστής)
3. Σύνθεση και οπτικοποίηση αποτελεσμάτων μοντέλου (μετεπεξεργαστής)

Να σημειωθεί πως σε περίπτωση που τα αποτελέσματα δείξουν ότι το μοντέλο δεν πληροί τις απαραίτητες προδιαγραφές αντοχής στο περιβάλλον λειτουργίας του

τότε γίνεται ανασχεδιασμός και επανέλεγχος με χρήση των εργαλείων CAD. Μερικά από τα πιο δημοφιλή εργαλεία λογισμικά *CAE* για τους διάφορους κλάδους της μηχανικής:

- *Ansys*
- *MATLAB*
- *Simulink*
- *Excel*
- *Abacus*

Συστήματα CAM

Τα συστήματα *CAM* (*Computer Aided Manufacturing*) αποσκοπούν στο έλεγχο παραγωγής με χρήση υπολογιστή ώστε να επιταχυνθούν οι παραγωγικές διαδικασίες και να παραχθούν ποιοτικότερα προϊόντα. Οι εργαλειομηχανές *CNC* είναι αυτές οι οποίες μεσολαβούν μεταξύ του υπολογιστή και των κατεργασιών παραγωγής. Τα λογισμικά *CAM* δέχονται ως είσοδο ψηφιακά κατασκευαστικά σχέδια και παράγουν ένα σύνολο εντολών που ελέγχουν τις λειτουργίες κατεργασίας της εργαλειομηχανής *CNC*. Πρόκειται λοιπόν για ένα εργαλείο προγραμματισμού και ελέγχου παραγωγής.

4. Τεχνολογίες 3D σάρωσης

Η τεχνολογία της τρισδιάστατης σάρωσης έχει σκοπό την αντιγραφή και αποτύπωση ενός φυσικού αντικειμένου ή χώρου σε ηλεκτρονική μορφή αναγνώσιμη από ειδικά λογισμικά *CAD* [δ7]. Μέσω αισθητηρίων οργάνων μέτρησης απόστασης και με διάφορες άλλες φωτογραφικές τεχνικές καταγράφονται δειγματοληπτικά σημεία συντεταγμένων X, Y, Z της επιφάνειας του αντικειμένου και αποθηκεύονται σε αρχείο ηλεκτρονικής μορφής. Το συσσωμάτωμα όλων αυτών των σημείων αναφέρεται ως «νέφος σημείων» όπου στην συνέχεια υπολογίζεται η επιφάνεια και παράγεται μία πολυγωνική αναπαράσταση πλέγματος της επιφάνειας. Υπάρχουν διάφορες τεχνικές και μεθοδολογίες για την αντιγραφή ενός αντικειμένου η καθεμία με τα πλεονεκτήματα και τα μειονεκτήματα της.

Φωτογραμμετρία

Η φωτογραμμετρία είναι η τεχνολογία απόκτησης γεωμετρικών πληροφοριών σχετικά με τα φυσικά αντικείμενα και το περιβάλλον μέσω της διαδικασίας εγγραφής, μέτρησης και ερμηνείας φωτογραφιών [δ8]. Η φωτογραμμετρία εμφανίστηκε στα μέσα του 19ου αιώνα , σχεδόν ταυτόχρονα με την εμφάνιση της ίδιας της φωτογραφίας. Υπάρχουν πολλές παραλλαγές της φωτογραμμετρίας και ένα παράδειγμα είναι η εξαγωγή τρισδιάστατων μετρήσεων από δισδιάστατα δεδομένα που παρέχουν οι φωτογραφίες. Ένα άλλο παράδειγμα είναι η απόσταση μεταξύ δύο σημείων που βρίσκονται σε επίπεδο παράλληλο προς το επίπεδο φωτογραφικής εικόνας μπορεί να προσδιοριστεί μετρώντας την απόστασή τους στην εικόνα σε εικονοστοιχεία, από την στιγμή που είναι γνωστή η κλίμακα της εικόνας. Επιπλέον είναι δυνατή η εξαγωγή χρωματικών πληροφοριών και τιμών που αντιπροσωπεύουν ποσότητες όπως το μέτρο της διάχυτης ανάκλασης και της ηλιακής ακτινοβολίας.

Μια ειδική περίπτωση είναι η στερεοφωτογραμμετρία η οποία αποσκοπεί στην εκτίμηση των τρισδιάστατων συντεταγμένων σημείων της επιφάνεια ενός αντικείμενου με βάση την σύνθεση και ερμηνεία επικαλυπτόμενων φωτογραφιών από διαφορετικές οπτικές γωνίες όπως φαίνεται στην **Εικόνα 1.1Εικόνα 1.1**. Ειδικά

λογισμικά ανιχνεύουν μεταξύ των φωτογραφιών κοινά σημεία στον χώρο και έπειτα με εφαρμογή αναλυτικής γεωμετρίας και τριγωνοποίησης παράγουν το ψηφιακό αρχείο. Οι πιο εξελιγμένοι αλγόριθμοι μπορούν να εκμεταλλευτούν άλλες πληροφορίες σχετικά με τη σκηνή που είναι γνωστή εκ των προτέρων όπως για παράδειγμα συμμετρίες. Γι αυτό τον λόγο κατά την διαδικασία λήψης φωτογραφιών προτιμάται να γίνεται με το αντικείμενο να παραμένει σταθερό και οι λήψεις να γίνονται περιμετρικά αυτού με μετακίνηση του μέσου εγγραφής.

Εικόνα 1.1: Παράδειγμα στερεοφωτογραμμετρίας. [ε1]

Πλεονεκτήματα:

- Οικονομία: Χρειάζεται μια κάμερα και το κατάλληλο λογισμικό επεξεργασίας των φωτογραφιών
- Ευκολία: Η λήψη φωτογραφιών γίνεται άμεσα χωρίς την χρήση ειδικών μηχανισμών.
- Το ψηφιακό αρχείο περιέχει χρωματικές πληροφορίες.

Μειονεκτήματα:

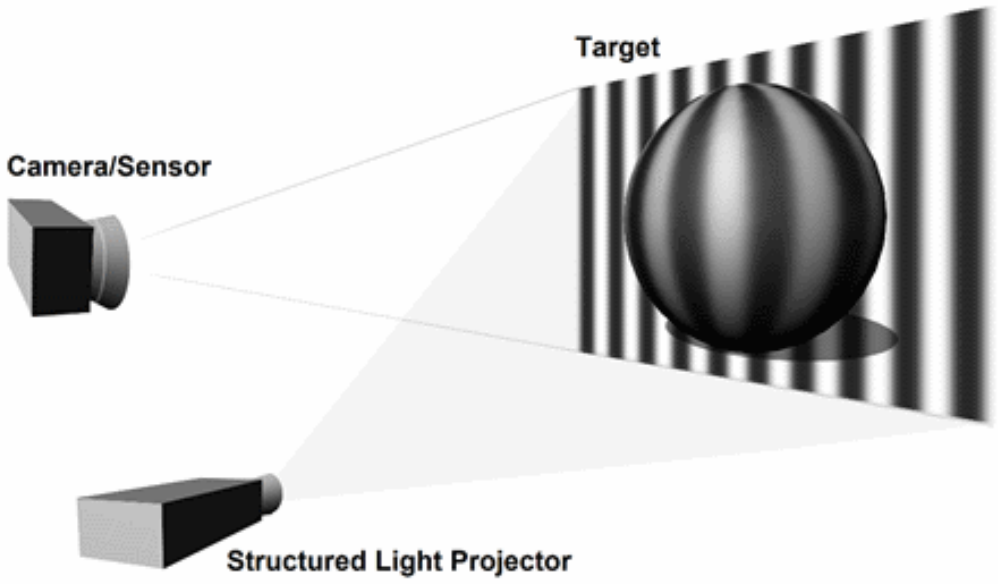
- Χρειάζεται μια σχετική εμπειρία ώστε να γίνει λήψη από όλες τις απαραίτητες οπτικές γωνίες με την κατάλληλη επικάλυψη.
- Ακρίβεια: Είναι συχνό το φαινόμενο θορύβου και σφαλμάτων στο ψηφιακό μοντέλο.
- Η ψηφιακή αναπαράσταση συνήθως βρίσκεται σε κλίμακα και όχι στις πραγματικές διαστάσεις. Θα πρέπει να γίνει περαιτέρω επεξεργασία για να διορθωθούν οι αναλογίες.
- Ταχύτητα: Η λήψη φωτογραφιών γίνεται χειροκίνητα. Τα λογισμικά χρησιμοποιούν αρκετούς πόρους των ηλεκτρονικών υπολογιστών και χρειάζονται αρκετές ώρες για να εξάγουν τα αποτελέσματα.

Κύριες εφαρμογές:

- Αποτύπωση ιστορικών αντικειμένων.
- Χαρτογράφηση με εναέριες λήψης σε απροσπέλαστες περιοχές.
- Αρχιτεκτονική.
- Αστυνομικές αρχές.
- Τέχνη.

Δομημένου φωτός

Η προβολή μιας στενής λωρίδας φωτός επάνω σε μια τρισδιάστατη επιφάνεια η οποία φαίνεται παραμορφωμένη από άλλες οπτικές γωνίες από αυτήν του προβολέα μπορεί να χρησιμοποιηθεί για την γεωμετρική ανακατασκευή του σχήματος της επιφάνειας [δ9]. Μια ταχύτερη και πιο ευέλικτη μέθοδος είναι η προβολή σχεδίων που αποτελούνται από πολλές ρίγες ταυτόχρονα καθώς αυτό επιτρέπει την ταυτόχρονη απόκτηση πλήθους δειγμάτων. Κοιτάζοντας από διαφορετικές οπτικές γωνίες το προβαλλόμενο μοτίβο εμφανίζεται παραμορφωμένο λόγω του επιφανειακού σχήματος του αντικειμένου. Η μέτρηση της μετατόπισης των λωρίδων επιτρέπει την ακριβή ανάκτηση των τρισδιάστατων συντεταγμένων στην επιφάνεια του αντικειμένου.



Εικόνα 1.2: Σάρωση δομημένου φωτός. [ε2]

Έχουν καθιερωθεί δύο μέθοδοι δημιουργίας μοτίβων λωρίδων φωτός, αυτή της προβολής δομημένου φωτός και προβολή λέιζερ. Η μέθοδος προβολής λέιζερ λειτουργεί με δύο πλατιά μπροστινά μέτωπα δέσμης λέιζερ που οδηγεί σε ισοδύναμα μοτίβα γραμμής. Μπορούν να επιτευχθούν διάφοροι αριθμοί παράλληλων γραμμών μοτίβων αλλάζοντας τη γωνία προβολής. Η μέθοδος επιτρέπει την ακριβή και εύκολη δημιουργία πολύ λεπτών σχεδίων με απεριόριστο βάθος πεδίου. Τα μειονεκτήματα είναι το υψηλό κόστος εφαρμογής και ο θόρυβος από τις ανακλαστικές επιφάνειες.

Η μέθοδος προβολής δομημένου φωτός χρησιμοποιεί λειτουργεί σαν βιντεοπροβολέας όπως φαίνεται στην **Εικόνα 1.2**. Τα μοτίβα δημιουργούνται συνήθως περνώντας φως μέσω ενός ψηφιακού διαμορφωτή χωρικού φωτός , που βασίζεται συνήθως σε μία από τις τρεις πλέον διαδεδομένες τεχνολογίες ψηφιακής προβολής, τους μεταδοτικούς υγρούς κρυστάλλους, τους ανακλαστικούς υγρούς κρυστάλλους σε πυρίτιο (*LCOS*) και την ψηφιακή επεξεργασία φωτός (*DLP*) οι οποίοι έχουν διάφορα συγκριτικά πλεονεκτήματα και μειονεκτήματα. Το μοτίβο φωτός συνήθως αποτελείται από ρίγες με περιοχές έντονου λευκού ή μπλε φωτός και περιοχές σκίασης.

Ένα τυπικό συγκρότημα μέτρησης σάρωσης και για τις δύο παραπάνω περιπτώσεις αποτελείται από έναν προβολέα και τουλάχιστον μία κάμερα. Σε πολλές εφαρμογές χρησιμοποιούνται δύο κάμερες στις αντίθετες πλευρές του προβολέα.

Πλεονεκτήματα

- Ταχύτητα: Το μοτίβο φωτός εφαρμόζεται σε όλη την επιφάνεια του αντικειμένου έτσι γίνεται ταυτόχρονη μέτρηση πολλών σημείων της επιφάνειας χωρίς να απαιτούνται λήψεις από πολλές οπτικές γωνίες
- Ακρίβεια: Καθώς είναι γνωστές όλες οι θέσεις των οργάνων σε σχέση με το αντικείμενο ο τριγωνισμός υπολογίζει τα σημεία με μεγάλη ακρίβεια και σωστή κλίμακα.

Μειονεκτήματα

- Σάρωση σχετικά μικρών αντικειμένων και σε κοντινές αποστάσεις
- Ευαίσθητο στις περιβαλλοντικές συνθήκες: Η ανάγκη για περιοχές σκίασης δημιουργεί πρόβλημα όταν υπάρχουν εξωτερικές πηγές φωτός με αποτέλεσμα το λογισμικό να μην μπορεί να ανιχνεύσει και να μετρήσει τα σημεία.
- Χρήση μόνο σε ματ επιφάνειες. Η αρχή λειτουργίας αυτής της μεθόδου βασίζεται στην καθαρότητας του μοτίβου, οποιαδήποτε αλληλεπίδραση φωτεινών και σκιασμένων περιοχών οδηγεί σε απρόσμενα αποτελέσματα και απώλεια ακρίβειας. Για την αντιμετώπιση αυτού του προβλήματος συνήθως εφαρμόζεται βαφή στην επιφάνεια.

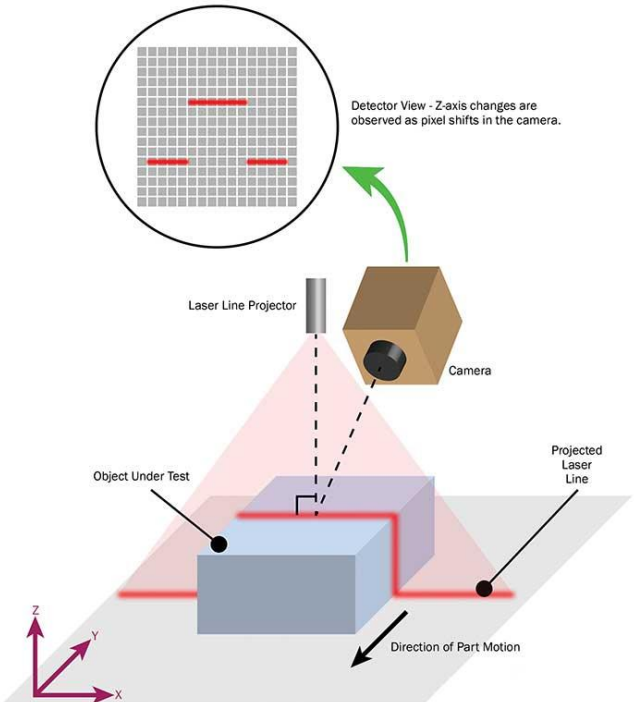
Χρήσεις και εφαρμογές

- Αντιγραφή αντικειμένων.
- Ογκομέτρηση στερεών αντικειμένων πολύπλοκης γεωμετρίας
- Καταγραφή κινήσεων και ενσωμάτωση σε παιχνίδια ή ταινίες επαυξημένης πραγματικότητας.
- Τέχνη και μόδα.
- Αποτύπωση ανθρώπινων μελών σε εφαρμογές ιατρικής.
- Συστήματα αναγνώρισης προσώπου.
- Συστήματα ανίχνευσης εμποδίων σε μη επανδρωμένα αεροσκάφη.
- Εκπαίδευση.

Τριγωνοποίησης δέσμης laser

Οι σαρωτές λέιζερ τρισδιάστατης τριγωνοποίησης χρησιμοποιούν γενικά λέιζερ ημιαγωγών κυρίως λόγω του χαμηλού κόστους και του μικρού τους μεγέθους

[δ10]. Σε αυτή τη μέθοδο η ψηφιοποίηση ξεκινά με την εκπομπή μιας ευθύγραμμης ακτίνας λέιζερ που παραμορφώνεται όταν κοιτάζεται σε διαφορετική γωνία από αυτή της προβολής στην επιφάνεια του αντικείμενου όπως φαίνεται στην **Εικόνα 1.3**. Μέσω μιας κάμερας, ο τρισδιάστατος σαρωτής αναλύει την παραμόρφωση της γραμμής που εκπέμπεται από το λέιζερ στα ανάγλυφα του αντικειμένου προκειμένου να προσδιορίσει μέσω τριγωνομετρικών υπολογισμών τη θέση στο χώρο. Η γωνία που σχηματίζεται μεταξύ της κάμερας και της δέσμης του λέιζερ, η απόσταση από την κάμερα στο αντικείμενο και εκείνη της πηγής του λέιζερ προς το αντικείμενο καθώς και ο χρόνος περιστροφής του αντικειμένου μέχρι να σαρωθεί όλη η επιφάνεια του, είναι όλες οι παράμετροι που καθιστούν δυνατό τον προσδιορισμό των χωρικών συντεταγμένων σημείων της επιφάνειας του αντικειμένου.



Εικόνα 1.3: Σάρωση με δέσμη λέιζερ. [ε3]

Πλεονεκτήματα

- Ταχύτητα σάρωσης
- Οικονομικές συσκευές
- Μεγάλη ακρίβεια

Μειονεκτήματα

- Σάρωση μικρών αντικειμένων και σε μικρές αποστάσεις
- Χρήση μόνο σε ματ επιφάνειες. Η λύση σε γυαλιστερές και διάφανες επιφάνειες είναι η εφαρμογή βαφής η σκόνη κιμωλίας.
- Δεν είναι όσο ακριβείς του δομημένου φωτός
- Το laser είναι επιβλαβές για τα μάτια

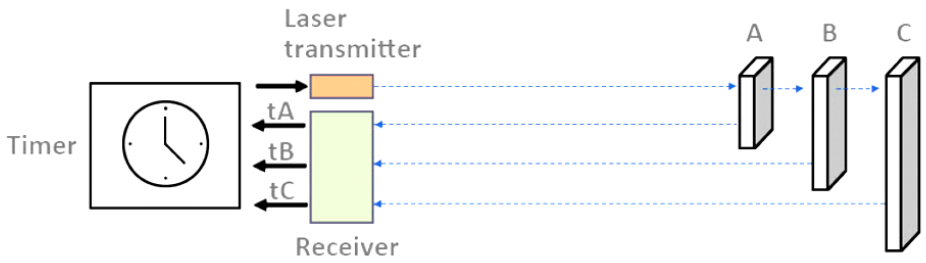
Εφαρμογές

- Αντιγραφή αντικειμένων
- Ογκομέτρήση στερεών αντικειμένων πολύπλοκης γεωμετρίας
- Καταγραφή κινήσεων και ενσωμάτωση σε παιχνίδια ή ταινίες επαυξημένης πραγματικότητας
- Αποτύπωση ανθρώπινων μελών σε εφαρμογές ιατρικής
- Ποιοτική επιθεώρηση σε γραμμές παραγωγής
- Εκπαίδευση

Αισθητήρων ToF

Η κάμερα χρόνου πτήσης (*ToF*) είναι ένα σύστημα πομπού-αισθητήρα εύρους που χρησιμοποιεί τεχνικές μέτρησης χρόνου πτήσης του φωτός για τον υπολογισμό της απόστασης μεταξύ της κάμερας και επιφάνειας μετρώντας τον χρόνο επιστροφής ενός τεχνητού σήματος φωτός που παρέχεται από λέιζερ ή LED [δ11] όπως φαίνεται και στο παράδειγμα στην **Εικόνα 1.4** για διαφορετικές αποστάσεις. Οι κάμερες-αισθητήρες χρόνου πτήσης που βασίζονται σε λέιζερ αποτελούν μέρος μιας ευρύτερης κατηγορίας ονομαζόμενη LIDAR στην οποία ολόκληρη η σκηνή καταγράφεται με κάθε παλμό λέιζερ. Τα συστήματα μπορούν να μετρήσουν από μερικά εκατοστά έως και χιλιόμετρα. Σε σύγκριση με άλλες μεθόδους σάρωσης οι κάμερες TOF έχουν μεγάλη ταχύτητα δειγματοληψίας σημείων. Υπάρχουν διάφορα είδη τεχνολογιών καμερών.

- Μέτρησης διαφοράς φάσης εκπεμπόμενου σήματος σε σχέση με το ανακλώμενο
- Μέτρησης χρονικής διάρκειας παλμών
- Μέτρηση χρονικής διάρκειας επιστροφής σήματος



Εικόνα 1.4: Παράδειγμα αισθητήρα χρόνου πτήσης. [ε4]

Πλεονεκτήματα

- Καλή ακρίβεια
- Σάρωση σε πραγματικό χρόνο
- Ταχύτητα σάρωσης
- Σάρωση μεγάλων αντικειμένων
- Εύρος αποστάσεων
- Σχετικά χαμηλό κόστος εξοπλισμού

- Χρωματικές πληροφορίες αντικειμένου με χρήση εξωτερικών καμερών
- Αντί για laser μπορεί να χρησιμοποιηθεί υπέρυθρο φως το οποίο δεν είναι επιβλαβές για τα μάτια

Μειονεκτήματα

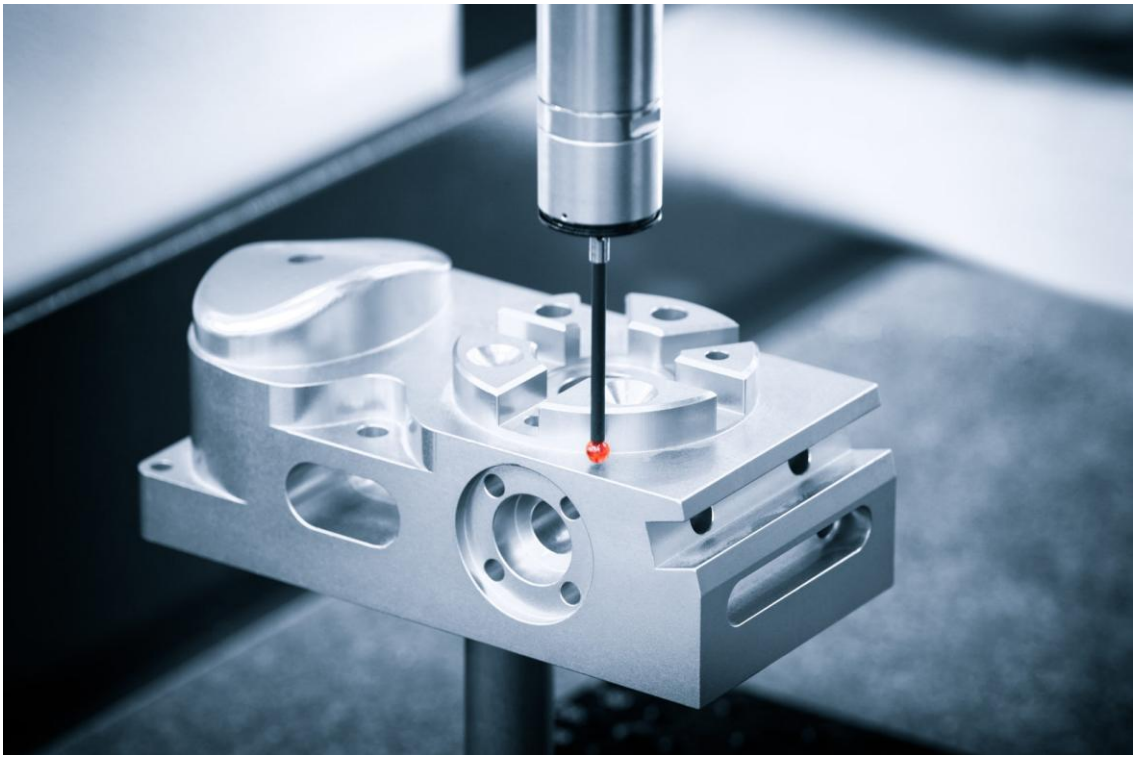
- Θόρυβος λόγω εξωτερικών πηγών υπέρυθρου φωτός
- Θόρυβος λόγω πολλαπλών ανακλάσεων σε γυαλιστερά αντικείμενα

Χρήσεις και εφαρμογές

- Συστήματα αποφυγής και εντοπισμού αντικειμένων
- Διεπαφής ανθρώπου και ηλεκτρονικών παιχνιδιών
- Σε γραμμές παραγωγής
- Αναπαράσταση γεωμορφολογίας και δωματίων

Επαφής

Οι σαρωτές αυτού του είδους όπως περιγράφονται και από το όνομα τους μετρούν σημεία της επιφάνειας των αντικειμένων μέσω αισθητήριων επαφής [δ12]. Οι κύριες διατάξεις που χρησιμοποιούνται είναι καρτεσιανά συστήματα ή βραχίονες έξι βαθμών ελευθερίας. Η διαδικασία μπορεί να γίνει χειροκίνητα ή αυτοματοποιημένα. Η καταγραφή της θέσης υπολογίζεται με βάση εξισώσεις αντίστροφης κινηματικής όταν η ακίδα επαφής αγγίξει την επιφάνεια του αντικειμένου. Ένα παράδειγμα αυτοματοποιημένης σάρωσης επαφής σε καρτεσιανό σύστημα φαίνεται στην **Εικόνα 1.5**.



Εικόνα 1.5: Καρτεσιανό σύστημα σάρωσης επαφής. [ε5]

Πλεονεκτήματα

- Μεγάλη ακρίβεια
- Σάρωση ανακλαστικών και διάφανων επιφανειών

Μειονεκτήματα

- Μικρή ταχύτητα σάρωσης
- Αλλοίωση των αντικειμένων λόγω απαίτησης επαφής
- Εφαρμογή μόνο σε μη ελαστικά αντικείμενα
- Σφάλματα μετρήσεων σε ελαστικές επιφάνειες
- Ακριβός εξοπλισμός

Κύριες χρήσεις

- Σάρωση μηχανολογικών εξαρτημάτων
- Ποιοτικός έλεγχος εξαρτημάτων

5. Επιλογή είδους σάρωσης

Για την βέλτιστη επιλογή του κατάλληλου είδους σαρωτή που θα κατασκευαστεί στα πλαίσια της παρούσας πτυχιακής θεωρούμε γνώμονα τα παρακάτω:

- Κόστος και διαθεσιμότητα υλικών.
- Ακρίβεια σάρωσης.
- Βαθμός αυτοματοποίησης διαδικασίας.
- Κατάλληλος για μηχανολογικά εξαρτήματα.

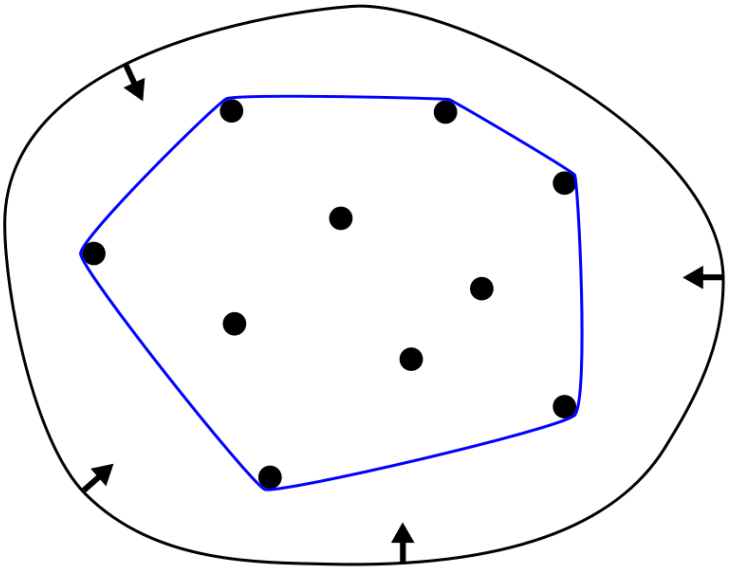
Με κριτήριο τα παραπάνω και σύμφωνα με όσα προαναφέρθηκαν για τις μεθόδους σάρωσης επιλέγεται ο τύπος με τον αισθητήρα ToF καθώς έχουν καλή σημειακή ακρίβεια με μικρό κόστος υλικών και ηλεκτρονικών μερών. Επιπλέον η διάταξη που θα κατασκευαστεί θα παρέχει αυτοματοποιημένη διαδικασία σάρωσης και θα μπορεί να χρησιμοποιηθεί ως έκθεμα στα εργαστήρια του ιδρύματος. Η χρήση του σαρωτή εκτός από εκπαιδευτική θα είναι και πρακτική καθώς θα είναι δυνατόν να γίνουν αντιγραφές αντικειμένων όπου δεν υπάρχουν τα αρχικά σχέδια, ογκομετρήσεις και επαναπαραγωγή αυτών με αντίστοιχες κατεργασίες σε τρισδιάστατους εκτυπωτές, φρέζες, τόρνους.

2.ΑΛΓΟΡΙΘΜΟΙ ΑΝΑΚΑΤΑΣΚΕΥΗΣ ΕΠΙΦΑΝΕΙΩΝ

Η ανοικοδόμηση της επιφάνειας είναι ένα πρόβλημα στον τομέα της υπολογιστικής γεωμετρίας που αφορά την αναδημιουργία επιφάνειας από διάσπαρτα σημεία δεδομένων που έχουν δειχθεί από άγνωστη επιφάνεια. Μέχρι πρότινος η πρωταρχική εφαρμογή αλγορίθμων ανακατασκευής επιφανείας εφαρμοζόταν στα γραφικά υπολογιστών, όπου τα φυσικά μοντέλα ψηφιοποιούνται σε τρεις διαστάσεις με σαρωτές εύρους λέιζερ ή μηχανικούς ανιχνευτές-ακίδες ψηφιοποίησης που εδράζουν επάνω σε βραχίονες ή καρτεσιανά συστήματα κίνησης [1]. Οι αλγόριθμοι επιφανειακής ανακατασκευής χρησιμοποιούνται για τη μετατροπή του συνόλου του νέφους σημείων σε ένα μοντέλο συρμάτινου πλέγματος το οποίο μπορεί να εμπεριέχει χρωματικές πληροφορίες και τοποθετούνται σε τρισδιάστατη σκηνή.

Η υπολογιστική γεωμετρία μπορεί να περιγραφεί ως το πεδίο που σχετίζεται με το σχεδιασμό και την εφαρμογή αλγορίθμων που χρησιμοποιούνται για την αποτελεσματική λύση των γεωμετρικών προβλημάτων.

Το κυρτό περίβλημα ενός συνόλου σημείων μπορεί να οριστεί ως το πολύγωνο που σχηματίζεται περιμετρικά των εξωτερικών σημείων όπως φαίνεται στην **Εικόνα 2.1** για τις δύο διαστάσεις [δ13]. Στις τρεις διαστάσεις μπορεί να θεωρηθεί ως το πολύεδρο το οποίο εμπεριέχει όλο το σύνολο σημείων με σύνορα τα πιο εξωτερικά σημεία. Από μαθηματικής άποψης ένα σύνολο Μ είναι κυρτό όταν κάθε ευθύγραμμο τμήμα που σχηματίζεται από κάθε ζεύγος σημείων x, y του Μ, ανήκει πλήρως στο Μ.

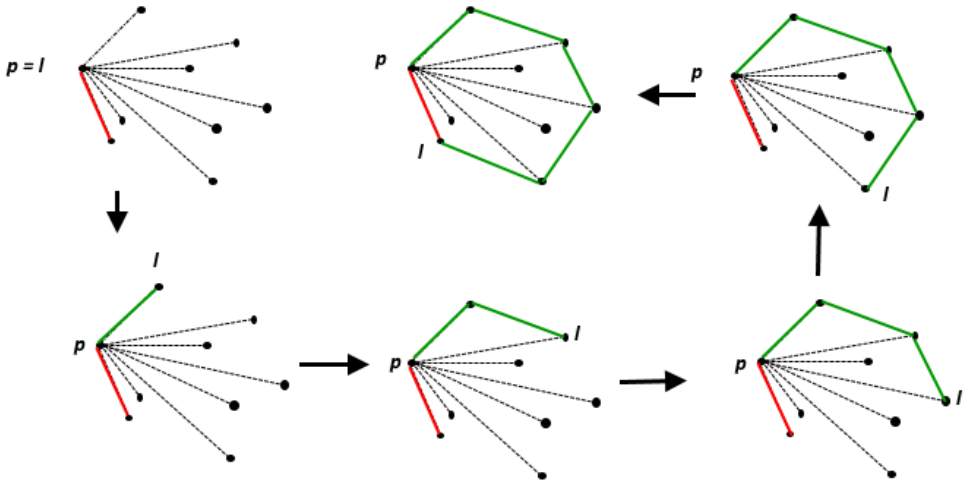


Εικόνα 2.1: Κυρτό περίβλημα στις δύο διαστάσεις. [ε6]

Δύο είναι τα κύρια προβλήματα που καλούνται οι αλγόριθμοι να λύσουν και περιλαμβάνει τον υπολογισμό των ακραίων σημείων-κορύφών του κυρτού περιβλήματος και ο υπολογισμός των συνόρων τους. Η παράσταση του συνόρου του κυρτού περιβλήματος εμπεριέχει περισσότερη πληροφορία από τις κορυφές. Φαινομενικά ο υπολογισμός των ακραίων σημείων φαίνεται πιο εύκολος αλλά στην πραγματικότητα δεν είναι. Παρακάτω αναφέρονται μερικοί διάσημοι αλγόριθμοι:

2.1 Αλγόριθμος Jarvis

Στην υπολογιστική γεωμετρία, ο αλγόριθμος αναδίπλωσης είναι ένας αλγόριθμος για τον υπολογισμό του κυρτού κύτους ενός δεδομένου συνόλου σημείων. Στην δισδιάστατη περίπτωση ο αλγόριθμος είναι επίσης γνωστός ως αλγόριθμος *Jarvis*, μετά τον *RA Jarvis*, ο οποίος τον δημοσίευσε το 1973 [2]. Η βασική ιδέα του συγκεκριμένου αλγόριθμου είναι ότι ξεκινάει από το αριστερότερο σημείο (ή σημείο με ελάχιστη τιμή συντεταγμένης *x*) και συνεχίζει τυλίγοντας τα σημεία σε δεξιόστροφη κατεύθυνση όπως φαίνεται και στην **Εικόνα 2.2**. Με αυτό τον τρόπο η εύρεση του επόμενου συνόρου γίνεται με βάση το τέλος του προηγούμενου όπως φαίνεται και στην παρακάτω εικόνα. Η πραγματική του απόδοση σε σύγκριση με άλλους αλγόριθμους κυρτού κύτους είναι ευνοϊκή όταν το ο αριθμός των σημείων είναι μικρός ή ο αριθμός των σημείων που ανήκουν στο κυρτό κύτος αναμένεται να είναι πολύ μικρό σε σχέση με τα συνολικά σημεία. Η αναγωγή του αλγόριθμου στις τρεις διαστάσεις περιγράφτηκε για πρώτη φορά από τους *Chand* και *Kapur* (1970) ως μια μέθοδο υπολογισμού κυρτών περιβλημάτων [3].



Εικόνα 2.2: Παράδειγμα εφαρμογής αλγόριθμου Jarvis. [ε7]

2.2 Αλγόριθμος Quickhull

Γενικά ο αλγόριθμος αυτός λειτουργεί αρκετά καλά, αλλά η επεξεργασία συνήθως καθυστερεί σε περιπτώσεις υψηλής συμμετρίας ή σημείων που βρίσκονται στην περιφέρεια ενός κύκλου. Ο αλγόριθμος διεκπεραιώνεται με την παρακάτω ακολουθία βημάτων στις δύο διαστάσεις του χώρου [4]:

1. Εύρεση των σημείων με ελάχιστες και μέγιστες συντεταγμένες x, καθώς αυτά θα είναι πάντα εσωτερικά του συνόλου. Εάν υπάρχουν πολλά σημεία με κοινό ελάχιστο / μέγιστο της τετμημένης x τότε χρησιμοποιούνται αυτά με ελάχιστο / μέγιστο της τεταγμένης y αντίστοιχα.
2. Χρησιμοποιείται η γραμμή που σχηματίζεται από τα δύο σημεία για να διαιρεθεί το σύνολο σε δύο υποσύνολα σημείων, τα οποία θα υποβληθούν σε επεξεργασία αναδρομικά.

3. Προσδιορισμός του σημείο στη μία πλευρά της γραμμής με τη μέγιστη απόσταση από αυτή. Αυτό το σημείο σχηματίζει ένα τρίγωνο με τις ακμές της γραμμής.
4. Τα σημεία που βρίσκονται εντός του τριγώνου δεν μπορούν να είναι μέρος του κυρτού κύτους και ως εκ τούτου μπορούν να αγνοηθούν στα επόμενα βήματα.
5. Επαναλαμβάνονται τα δύο προηγούμενα βήματα στις δύο γραμμές που σχηματίζονται από το τρίγωνο εκτός της αρχικής γραμμής.
6. Η διαδικασία συνεχίζεται μέχρι να μην απομείνουν άλλα σημεία και ο βρόγχος τελειώνει όταν τα επιλεγμένα σημεία αποτελούν το κυρτό κύτος όπως φαίνεται στην παρακάτω εικόνα.

Το πρόβλημα γίνεται πιο περίπλοκο στην περίπτωση υψηλών διαστάσεων, καθώς το κυρτό κύτος είναι κατασκευασμένο από πολλές όψεις αντί ευθυγράμμων τμημάτων. Ο αλγόριθμος πλέον πρέπει να καταγράψει το επίπεδο και τα ευθύγραμμα τμήματα που μοιράζονται μεταξύ τους και οι γειτονικές όψεις. Όποτε στην γενική περίπτωση για n διαστάσεις: [5]

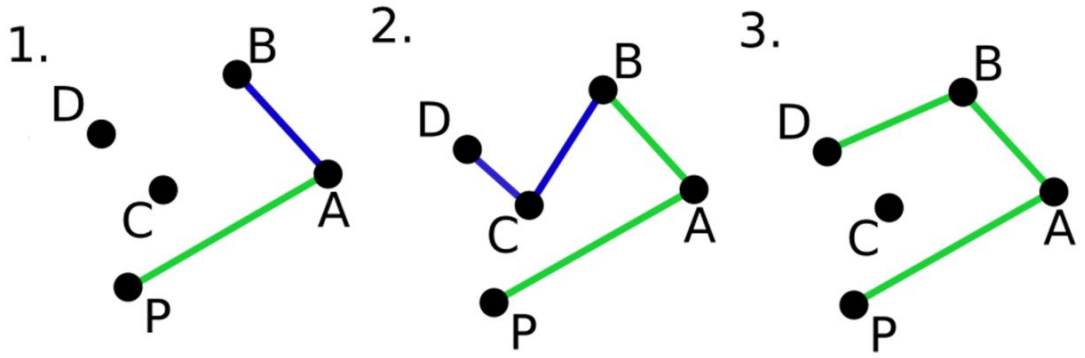
- Διαλέγονται *n + 1* βαθμοί από το σετ που δεν μοιράζονται κοινή όψη. Αυτό σχηματίζει ένα αρχικό κύτος με όψεις *Fs*.
- Για κάθε όψη *F* του σύνολο των όψεων *Fs* του κυρτού περιβλήματος, βρίσκονται όλα τα σημεία που είναι από «πάνω» της και δημιουργούνται καινούριες όψεις *FO* που σχετίζονται με το αρχικό επίπεδο *F* .
- Για κάθε όψη *F* που σχετίζεται με τις όψεις *FO* :

    1. Βρίσκεται το σημείο p με τη μέγιστη απόσταση από την F το οποίο προσθέτεται στο κύτος.
    2. Δημιουργεί μια ορατή ομάδα *V* και την αρχικοποιεί στο *F*. Επεκτείνετε το *V* σε όλες τις κατευθύνσεις για τις γειτονικές όψεις *Fv* έως ότου δεν φαίνονται άλλες όψεις από το *p.Fv*. Να είναι ορατό από το *p* σημαίνει ότι το *p* είναι πάνω από το *Fv*.
    3. Το όριο του *V* σχηματίζει το σύνολο των ακμών *H* .
    4. Ορίζεται *Fnew* το σύνολο των όψεων που δημιουργούνται από το *p* και τις κορυφογραμμές *H* .
    5. Για κάθε νέα όψη στο *Fnew*, εκτελείτε το βήμα (2) και αρχικοποιούνται ξανά τα εξωτερικά σύνολα *V*. Αυτή τη φορά ελέγχονται μόνο τα σημεία που βρίσκονται έξω από μια όψη στο *V* χρησιμοποιώντας τα εξωτερικά τους σύνολα V_i. αφού έχει γίνει επέκταση μόνο προς αυτή την κατεύθυνση.
    6. Διαγράφονται οι εσωτερικές όψεις στο *V* από το σύνολο *Fs* και προσθέτονται οι νέες όψεις στο *Fnew* από το *Fs* μέχρι να σαρωθεί όλο το *Fs*.

2.3 Αλγόριθμος Graham

Το όνομα το πήρε από τον *Ronald Graham* ο οποίος δημοσίευσε τον αρχικό αλγόριθμο το 1972 [6]. Ο αλγόριθμος βρίσκει όλες τις κορυφές του κυρτού κύτους που ταξινομούνται κατά μήκος των ορίων του και χρησιμοποιεί μια στοίβα για να ανιχνεύει και να αφαιρεί τις κοιλότητες στο όριο αποτελεσματικά όπως φαίνεται στην **Εικόνα 2.3**.

1. Το πρώτο βήμα του αλγόριθμου είναι να βρεθεί το σημείο με την μικρότερη τεταγμένη *y*. Εάν η χαμηλότερη τεταγμένη *y* υπάρχει σε περισσότερα από ένα σημεία του συνόλου τότε επιλέγεται το σημείο με τη χαμηλότερη τετμημένη *x*. Το σημείο αυτό καλείται *P* . Αυτό το βήμα εκτελείται *n* φορές όπου *n* είναι ο αριθμός των σημείων του νέφους.
2. Στη συνέχεια, το σύνολο των σημείων ταξινομείται με αύξουσα σειρά της γωνίας και του σημείου *P* που διαγράφει με τον άξονα *x*. Οποιοσδήποτε αλγόριθμος ταξινόμησης γενικού σκοπού είναι κατάλληλος για αυτό όπως για παράδειγμα ο αλγόριθμος της φυσαλίδας. Η ταξινόμηση της γωνίας δεν απαιτεί τον υπολογισμό της γωνίας αλλά είναι δυνατή η χρήση οποιασδήποτε συνάρτησης της. Το συνημίτονο υπολογίζεται εύκολα χρησιμοποιώντας το εσωτερικό γινόμενο ή μπορεί να χρησιμοποιηθεί η κλίση της ευθείας. Εάν διακυβεύεται η αριθμητική ακρίβεια τότε η συνάρτηση σύγκρισης που χρησιμοποιείται από τον αλγόριθμο ταξινόμησης μπορεί να χρησιμοποιήσει το εξωτερικό γινόμενο για να προσδιορίσει σχετικές γωνίες.
3. Εν συνέχεια ο αλγόριθμος εξετάζει κάθε ένα από τα σημεία της ταξινομημένης σειράς και για κάθε σημείο καθορίζεται πρώτα εάν η τροχιά από τα δύο σημεία που προηγούνται αυτού του σημείου γίνεται αριστερόστροφα ή δεξιόστροφα. Στην δεξιόστροφη περίπτωση το δεύτερο προς το τελευταίο σημείο δεν είναι μέρος του κυρτού κύτους και αλλά εμπεριέχεται σε αυτό. Έπειτα γίνεται ο ίδιος προσδιορισμός για το ζεύγος του τελευταίου σημείου και των δύο σημείων που προηγούνται αμέσως του σημείου που βρέθηκε ότι ήταν εντός του κυρτού κύτους, επαναλαμβανόμενα έως ότου βρεθεί ένα αριστερόστροφο ζεύγος. Οπότε ο αλγόριθμος κινείται στο επόμενο σημείο στο σύνολο των σημείων της ταξινομημένης συστοιχίας μείον τυχόν σημεία που βρέθηκαν να βρίσκονται μέσα στο κύτος τα οποία δεν χρειάζεται να επανεξεταστούν.



Εικόνα 2.3: Αναπαράσταση αλγόριθμου Graham. [ε8]

Να σημειωθεί ότι η ίδια βασική ιδέα λειτουργεί επίσης εάν η είσοδος ταξινομείται σε συντεταγμένες *x* αντί για γωνία και το κυρτό κύτος υπολογίζεται σε δύο στάδια παράγοντας το άνω και το κάτω μέρος του κύτους αντίστοιχα. Αυτή η τροποποίηση επινοήθηκε από τον *AM Andrew* και είναι γνωστός ως Αλγόριθμος Μονοτονικής Αλυσίδας του *Andrew*. Έχει τις ίδιες βασικές ιδιότητες με τη σάρωση του *Graham*.

2.4 Incremental αλγόριθμος

Μια κρίσιμη διάκριση από προηγούμενους αλγόριθμους επιφανειακής ανακατασκευής είναι ότι η τρέχουσα μέθοδο μπορεί να ενσωματωθεί σε έναν αυθαίρετο χώρο *Hilbert* με περισσότερες από τις τρεις διαστάσεις του Ευκλείδειου χώρου. Ο τρέχων αλγόριθμος είναι «τοπολογικά εγγενής» καθώς εξαρτάται μόνο από την γεωμετρική πολλαπλότητα του ίδιου και όχι του χώρου που βρίσκεται. Πρακτικά ο αλγόριθμος μπορεί να ανακατασκευάσει πολλές επιφάνειες που δεν δύναται να ανακατασκευαστούν χρησιμοποιώντας τους γνωστούς αλγόριθμους, όπως για παράδειγμα μη προσανατολισμένες επιφάνειες όπως ως το μπουκάλι *Klein* [7].

Το κόστος αυτής της γενίκευσης είναι η αδυναμία να προσδιοριστεί με σιγουριά η λειτουργία του αλγόριθμου σε ανώτερες διαστάσεις. Προς το παρόν δεν αποδεικνύονται σημαντικά αποτελέσματα σχετικά με την εγκυρότητα του καθώς είναι αδύνατο να ελεγθεί. Παρόλα αυτά παρουσιάζεται ως ένας αλγόριθμος που λειτουργεί καλά σε ένα πειραματικό στάδιο.

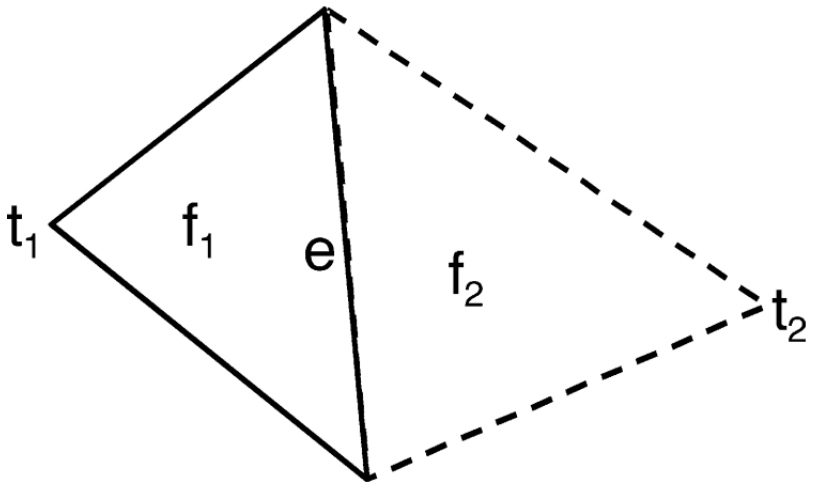
Σε κάθε επανάληψη η επιφάνεια αναπτύσσεται προσθέτοντας ένα μόνο τρίγωνο. Ένα σημαντικό ζήτημα που σχετίζεται με έναν τέτοιο αυξητικό αλγόριθμο είναι το ακόλουθο: πώς είναι δυνατόν να αποφευχθεί η επιλογή τριγώνων που οδηγούν σε αυτοδιασταυρώσεις των επιφανειών; Το ζήτημα αντιμετωπίζεται εκτελώντας όλους τους υπολογισμούς τοπικά σε ένα επίπεδο που προσεγγίζει τους κοντινούς εφαπτομενικούς χώρους του σχήματος. Πάλι, να σημειωθεί ότι η έμφαση αυτής της εργασίας είναι σε μεγάλο βαθμό πειραματική και δεν προσφέρονται αποδείξεις και ο αλγόριθμος έχει βάση στην άποψη της γεωμετρικής και τοπολογικής μαθηματικής διαίσθησης.

Η στοιχειώδη φύση του είναι να αυξάνει την επιφάνεια προσθέτοντας μία απλή έδρα κάθε φορά. Σε κάθε επανάληψη ο αλγόριθμος επιλέγει ένα άκρο e το οποίο άκρο ανήκει σε μία έδρα *f1*. Δεδομένου ότι η επιφάνεια θεωρείται ότι δεν έχει όριο κάθε άκρη πρέπει να ανήκει ακριβώς σε δύο έδρες όπως φαίνεται στην **Εικόνα 2.4**. Ο στόχος είναι να βρεθεί η δεύτερη έδρα *f2* που μοιράζεται το κοινό άκρο *e* με την έδρα *f1*. Αυτός είναι ο τρόπος με τον οποίο ο αλγόριθμος είναι σταδιακός, και απεικονίζεται στην εικόνα. Με την πρόσθεση της *f2* επιφάνειας το άκρο που θα είναι πλήρως ορισμένο από την τομή των δύο επιπέδων. Το νέο πολύγωνο που δημιουργείται πλέον μπορεί έχει καινούριες άκρες. Ο αλγόριθμος τερματίζεται όταν δεν υπάρχουν άκρα τα οποία δεν θα ορίζονται πλήρως.

Η βασική αλγοριθμική ερώτηση που προκύπτει είναι πως για μια δεδομένη ακμή *e* επιλέγεται η έδρα *f2*; Παρακάτω παρουσιάζονται τα τρία στάδια:

1. Φιλτράρισμα: Πριν από την προσθήκη του επιπέδου *f2* υπάρχει μόνο η πολλαπλότητα *K* και πρέπει να διασφαλιστεί ότι μετά την προσθήκη του *f2* θα εξακολουθεί να διατηρείτε η ίδια. Οι μόνες επιφάνεια που μπορούν να τέμνουν οποιαδήποτε πιθανή επιφάνεια f_2 είναι αυτές που είναι γειτονικά στο άκρο *e.* Ως αποτέλεσμα είναι να φιλτράρονται και τα σημεία και το σύμπλεγμα K έτσι ώστε να παρθεί το υποσύνολο σημείων *Γ* και το υποσύμπλεγμα *L* τα οποία είναι γειτονικά στην ακμή *e*. Επειδή δεν είναι απλό να χρησιμοποιηθεί ένα κριτήριο απόστασης για να εντοπιστούν τα γειτονικά στο *e* καθώς τα σημεία μπορεί να είναι ανομοιόμορφα αντί αυτού χρησιμοποιείται εφαπτομενική συνθήκη.

2. Προβολή: Μόλις υπολογιστεί η υποπολλαπλότητα που βρίσκεται γειτονικά του αιωρούμενου άκρου *e*, τότε προβάλλετε αυτή στο επίπεδο. Ο λόγος που αυτό είναι εφικτό αυτό είναι γιατί η υποπολλαπλότητα τοπολογικά είναι ένα σύνολο από 2-μπάλες, 1-μπάλα και 0-μπάλες. Επί πλέον με βάση τον τρόπο που πραγματοποιείται το φιλτράρισμα για να ληφθεί η υποπολλαπλότητα *L*, η προβολή της είναι κοντά σε μια ισομετρία των φιλτραρισμένων σημείων. Ως αποτέλεσμα, το πρόβλημα εύρεσης της έδρας *f2* μειώνεται στο πρόβλημα της προσθήκης ενός τριγώνου σε έναν μερικώς ολοκληρωμένο τριγωνισμό του επιπέδου.
3. Τριγωνοποίηση: Ο στόχος είναι να βρεθεί μια κορυφή t_2 στο προβαλλόμενο επίπεδο το οποίο όταν συνδυάζεται με το άκρο *e*, να παράγει την κατάλληλη έδρα f_2. Αυτή η κορυφή t_2 (που ανήκει στο φιλτραρισμένο υποσύνολο δειγμάτων *Γ*) μπορεί είτε να είναι μια απομονωμένη κορυφή, ή μπορεί να ανήκει σε ένα άκρο ή ένα τρίγωνο. Σε κάθε περίπτωση, το κλειδί είναι για το νέο τρίγωνο f_2 να μην τέμνει οποιοδήποτε άλλη έδρα που ανήκει στο υποσύνολο *L*.



Εικόνα 2.4: Δύο έδρες μοιράζονται μία κοινή ακμή. [ε9]

2.5 Διαιρεί και βασίλευε

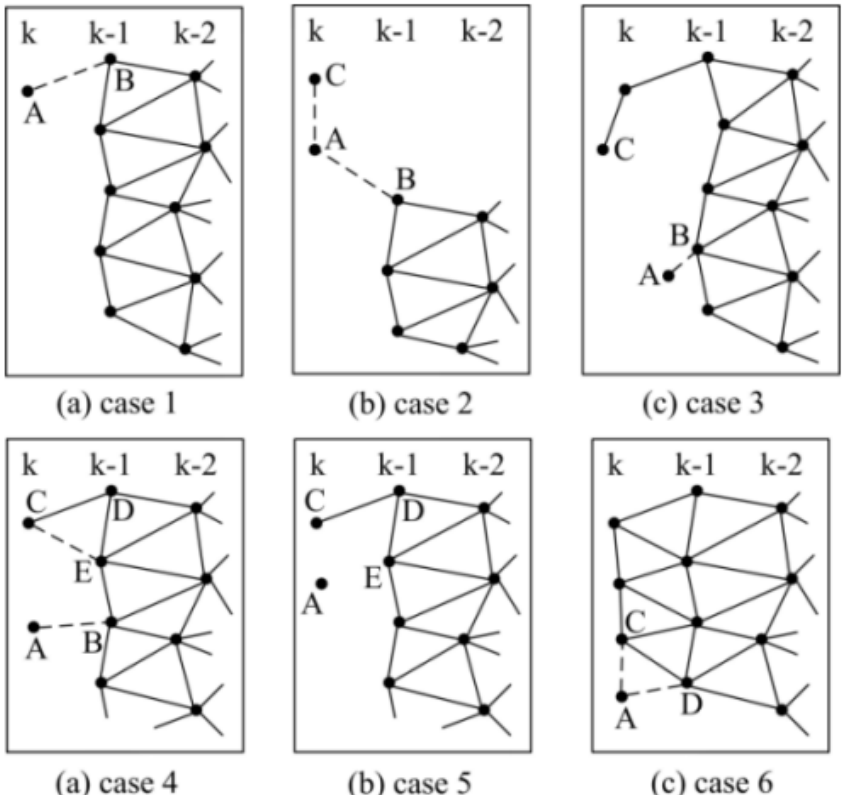
Σε αυτή τη μέθοδο αντί να υπολογιστεί η επιφάνεια από τα αρχικά σημεία συλλογής που προκύπτουν συνήθως από σάρωση λέιζερ, αρχικά γίνεται επεξεργασία των σημείων δεδομένων του νέφους γραμμή προς γραμμή σάρωσης και σημείο προς σημείο σύμφωνα με τη σειρά πρόσβασης της σάρωσης κάθε γραμμής και κάθε χωρικής θέσης του διάσπαρτου νέφους. Κατά τη διαδικασία της επεξεργασίας το νέα εισαγόμενα σημεία χωρίζονται σε τέσσερις κατηγορίες [8]:

- σε αραιό σημείο
- σε σημείο πυκνότητας
- σε σημείο ανάπτυξης
- και σε σημείο επιστροφής

Αυτό γίνεται σύμφωνα με τη σχετική σχέση θέσης μεταξύ του νέου σημείου εισαγωγής και της παραγόμενης επιφάνειας. Ύστερα μπορούν να εφαρμοστούν διάφοροι αλγόριθμοι δημιουργίας επιφανειών για την επεξεργασία του νέου σύννεφου σημείων και να παραχθεί η τελική τρισδιάστατοι επιφάνεια χρησιμοποιώντας τα χαρακτηριστικά κατανομής σημείων διάσπαρτου κάθε περίπτωσης. Η διαδικασία βημάτων για το φιλτράρισμα του νέφους περιλαμβάνει:

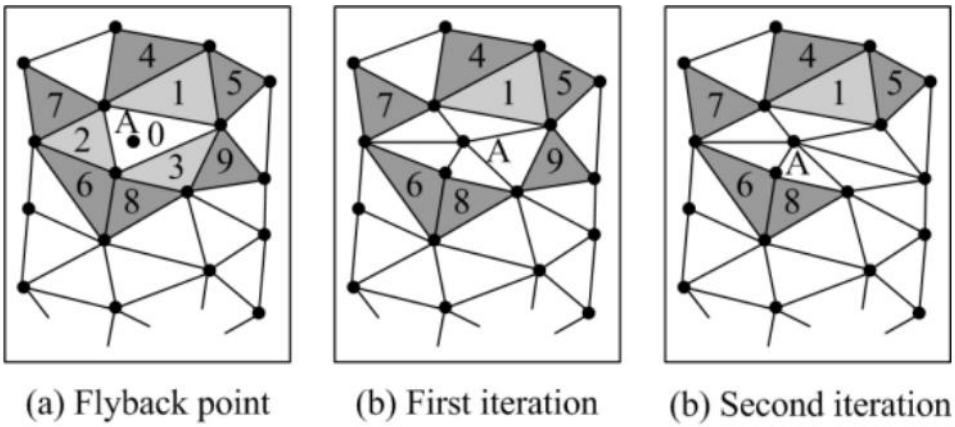
1. Προεπεξεργασία νέφους σημείων: Χρησιμοποιείται η μέση τιμή τοπικών σημείων για την εκτέλεση μέσου φιλτραρίσματος στη σάρωση δεδομένων νέφους σημείου. Η κυματοειδής περιοχές του αντικειμένου αντιμετωπίζονται με βελτιωμένη καμπυλότητα και οι σχετικά επίπεδες περιοχές με ομοιόμορφη δειγματοληψία. Η διανυσματική γωνία που σχηματίζεται από το φιλτραρισμένο σημείο και τα δύο διάσπαρτα αρχικά σημεία πριν και μετά ορίζει την καμπυλότητα. Εάν η καμπυλότητα των αρχικών σημείων προς το φιλτραρισμένο σημείο είναι μικρότερη από μία ελάχιστη τιμή αυτό σημαίνει ότι η περιοχή είναι καμπύλη, αλλιώς το σημείο βρίσκεται σε σχετικά επίπεδη περιοχή και πραγματοποιείται ομοιόμορφη δειγματοληψία. Στην ομοιόμορφη δειγματοληψία εάν η απόσταση από το τρέχον σημείο στο προηγούμενο σημείο είναι μεγαλύτερη από ένα όριο μέγιστης απόστασης τότε σημείο θα διατηρηθεί διαφορετικά το σημείο διαγράφεται.
2. Τοποθέτηση των εισαχθέντων σημείων: Πρέπει να γίνει βέβαιο η σχέση τοποθεσίας μεταξύ κάθε αρχικού διάσπαρτου σημείου και της αρχικής επιφάνειας, το οποίο σημαίνει ότι πρέπει να βρεθεί το σημείο στην επιφάνεια που είναι πιο κοντά στο πρόσφατο σημείο. Ένας τοπικός αλγόριθμος αναζήτησης με βάση την προσαρμοστική τμηματοποίηση του νέφους σύμφωνα με τα μεταβαλλόμενα χαρακτηριστικά της περιοχής των σημείων του νέφους. Στον αλγόριθμο, όλα διάσπαρτα σημεία προβάλλονται πρώτα στο επίπεδο *XY*, η περιοχή προβολής της πρώτης γραμμής σάρωσης τοποθετεί τα σημεία σε ίσα διαστήματα. Στη συνέχεια, κάθε νέο σημείο που εισάγεται στην καινούρια γραμμή σάρωσης θα υποβληθεί σε συνεχή επεξεργασία, εάν η θέση προβολής του προσφάτως εισαχθέντος σημείου είναι εκτός του εύρους του αρχικού διαστήματος τμηματοποίησης, τότε αυξάνεται ο αριθμός των διαστημάτων τμηματοποίησης και η αρίθμησή τους. Επομένως μέχρι στιγμής, έχουν αντιστοιχηθεί όλα τα δεδομένα του νέφους εισαγωγής σε πολλαπλά τμήματα με ίσες αποστάσεις διαστήματα με τον αλγόριθμο τμηματοποίησης νέφους προσαρμοστικού σημείου. Όταν χρειαστεί να αναζητηθεί ένα εγγύς σημείο κάποιου σημείου τότε μόνο το διάστημα τμηματοποίησης στο οποίο είναι το σημείο και η εγγύς τμηματοποίηση αναζητούνται. Δεδομένου ότι η πυκνότητα των δεδομένων σημείων νέφους που λαμβάνονται από το σύστημα σάρωσης είναι σχετικά ομοιογενές, και ο αριθμός των διάσπαρτων σημείων είναι βασικά ο ίδιος, ο χρόνος υπολογισμού δεν αυξάνεται με την αύξηση του συνολικού αριθμού των διασκορπισμένων σημείων το την διαδικασία αναζήτησης.

3. Επεξεργασία πυκνών σημείων: Σε αυτή την περίπτωση γίνεται διαγραφή των πυκνών σημείων και μένει μόνο ένα στην περιοχή. Τότε το σημείο που απομένει θα μεταφερθεί στο καινούριο νέφος ως έχει εφόσον δεν αποτελεί την αρχή της γραμμής σάρωσης. Για να διασφαλιστεί η τοπική επιπεδότητα της τρισδιάστατης επιφάνειας στη βελτιστοποίηση του χωρικού τριγωνικού πλέγματος, της περιοχής των τριγώνων που αποτελούν τρισδιάστατη επιφάνεια πρέπει να είναι μικρή που ακόμα και σε κυματοειδή επιφάνεια να διατηρείται η καμπυλότητα στο σύνολο της αποτελούμενη από πολλές επίπεδες επιφάνειες. Για αυτόν τον λόγο εφαρμόζεται ένα κριτήριο βελτιστοποίησης τριγωνικού πλέγματος της μέγιστης διεδρικής γωνίας. Για τη βελτιστοποίηση του δύο γειτονικών τριγώνων εάν η διεδρική γωνία της μία πλευράς είναι μεγαλύτερη από την διεδρική της άλλης, τότε τα δύο τρίγωνα δεν χρειάζονται βελτιστοποίηση. Διαφορετικά αφαιρείται το ένα κοινό άκρο και ενώνονται δύο κορυφές του τριγώνου σχηματίζοντας ένα νέο βελτιστοποιημένο τρίγωνο. Μετά την δημιουργία του καινούριο πλέγματος επιφανείς αποτελούμενο από τρίγωνα, αλλάζει και το σύννεφο σημείων όπου κάθε σημείο ανήκει στην κορυφές των τριγώνων.
4. Επεξεργασία αραιών σημείων: Εάν το σημείο που εισήχθη πρόσφατα κρίνεται ως υπερβολικά αραιό σημείο αυτό δείχνει ότι δεν υπάρχει σωστό σημείο αντιστοίχισης κοντά σε αυτό το σημείο για να σχηματιστεί ένα τρίγωνο. Σε αυτό το στάδιο ελέγχεται εάν αυτό το σημείο είναι το πρώτο σημείο στη γραμμή σάρωσης και εάν το αποτέλεσμα είναι αληθές τότε το σημείο αποθηκεύεται απευθείας, διαφορετικά υπολογίζεται η απόσταση μεταξύ αυτού του σημείου και του εμπρόσθιου σημείου της ίδιας γραμμής σάρωσης. Εάν η απόσταση είναι μικρότερη από το μέγιστο όριο απόστασης τα δύο σημεία συνδέονται. Εάν η απόσταση είναι μεγαλύτερη από το μέγιστο όριο απόστασης τότε το σημείο αποθηκεύεται και ελέγχεται το επόμενο της ίδιας γραμμής σάρωσης με το μεθεπόμενο.
5. Επεξεργασία σημείων ανάπτυξης: Κατά τη διαδικασία επεξεργασίας εάν ένα σημείο *A* είναι το πρώτο σημείο μιας γραμμής σάρωσης τότε συνδέεται με ένα σημείο *B* (*case a*) όπως φαίνεται και στην **Εικόνα 2.5**, διαφορετικά ελέγχεται το σημείο *C* και κρίνεται αν είναι αραιό. Εάν το σημείο *C* είναι ένα υπερβολικά αραιό και η απόσταση μεταξύ του σημείου *A* και του σημείου *C* είναι μικρότερη από το μέγιστο όριο απόστασης τότε συνδέονται οι γραμμές *AB* και *AC* (*case b*). Εάν το σημείο *C* είναι ένα υπερβολικά αραιό σημείο και η απόσταση μεταξύ του σημείου *A* έως το σημείο *C* είναι μεγαλύτερη από το μέγιστο όριο απόστασης τότε συνδέονται το *A* και το σημείο *B* (*case C*). Εάν το *C* δεν είναι υπερβολικά αραιό σημείο και εάν η απόσταση μεταξύ του σημείου *A* έως το σημείο *C* είναι μεγαλύτερη από το ανώτατο όριο απόστασης, τότε το *A* και το *B* συνδέονται και το σημείο *C* υποβάλλεται σε επεξεργασία και θα βελτιστοποιηθεί ως το τελικό σημείο (*case d*). Εάν η απόσταση μεταξύ του σημείου *A* και του σημείου *C* είναι μικρότερη από το μέγιστο όριο απόστασης, τα τρίγωνα δημιουργούνται σύμφωνα με το κριτήριο του δίεδρου μεγιστοποίησης γωνίας.



Εικόνα 2.5: Επεξεργασία των σημείων ανάπτυξης. [ε10]

6. Επεξεργασία σημείων επιστροφής: Το σημείο επιστροφής *A* φαίνεται στην παρακάτω **Εικόνα 2.6**, προβάλλεται στο τρίγωνο *0* των παραγόμενων τριγώνων. Στη διαδικασία επανάληψης διαγράφονται και ανακατασκευάζονται τα γειτονικά τρίγωνα με το τρίγωνο *0* ως κέντρο. Στην πρώτη επανάληψη βελτιστοποιούνται τα αντίστοιχα τετράπλευρα του διαστήματος που αποτελούνται από τα σημεία *A* και τις τρεις κορυφές των τριγώνων 1, 2 και 3 χρησιμοποιώντας το κριτήριο μεγιστοποίησης γωνίας διεδρίας και το αποτέλεσμα της πρώτης επανάληψης είναι όπως φαίνεται στο σχήμα 6 (b). Τα καινούρια τρίγωνα πρέπει να βελτιστοποιηθούν και αυτά. Οπότε στη δεύτερη επανάληψη βελτιστοποιούνται τα γειτονικά τρίγωνα με το τρίγωνο που δημιουργήθηκε πρόσφατα και το αποτέλεσμα φαίνεται στο σχήμα 6 (c). Η επανάληψη τελειώνει όταν όλα τα γειτονικά τρίγωνα με το τελευταίο τρίγωνο που δημιουργήθηκε στην προηγούμενη επανάληψη δεν χρειάζεται να υποβληθούν σε επεξεργασία.



Εικόνα 2.6: Επεξεργασία σημείων επιστροφής. [ε10]

2.6 Τριγωνισμός Delaunay

Στα μαθηματικά και την υπολογιστική γεωμετρία ο τριγωνισμός *Delaunay* για ένα δεδομένο σύνολο *P* των διακεκριμένων σημείων σε ένα επίπεδο είναι ένας τριγωνισμός έτσι ώστε κανένα σημείο του *P* να είναι μέσα στο περιγεγραμμένο κύκλο οποιουδήποτε τριγώνου. Ο τριγωνισμός *Delaunay* μεγιστοποιεί την ελάχιστη γωνία όλων των γωνιών των τριγώνων κατά την τριγωνοποίηση. Ο αλγόριθμος πήρε το όνομά του από τον *Boris Delaunay* για το έργο του σε αυτό το θέμα από το 1934.

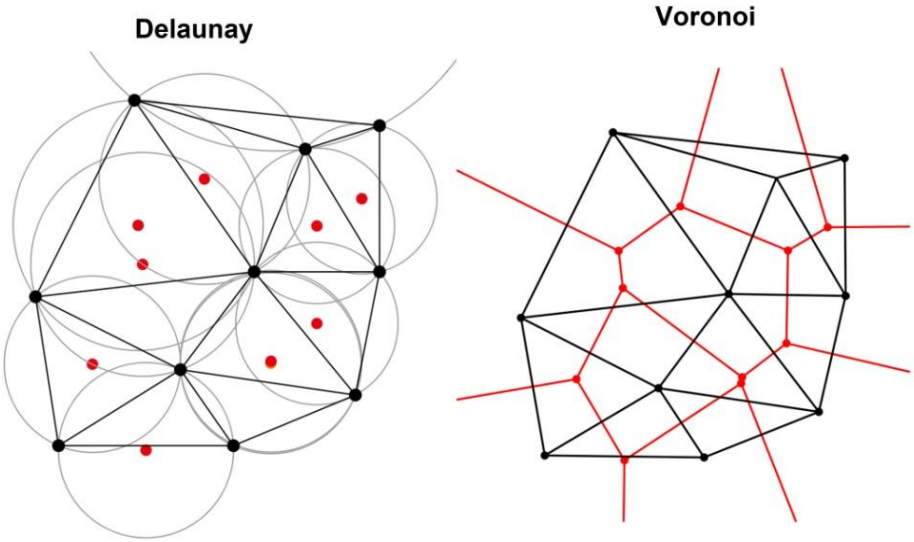
Για συνευθειακά σημεία δεν υπάρχει τριγωνισμός *Delaunay*. Για τέσσερα ή περισσότερα σημεία στον ίδιο κύκλο για παράδειγμα οι κορυφές ενός ορθογωνίου, ο τριγωνισμός *Delaunay* δεν είναι μοναδικός αλλά καθένας από τους δύο πιθανούς τριγωνισμούς που χωρίζουν το τετράγωνο σε δύο τρίγωνα ικανοποιεί την συνθήκη *Delaunay*. Λαμβάνοντας υπόψη τις περιορισμένες σφαίρες, η έννοια του τριγωνισμού *Delaunay* εκτείνεται σε τρεις και υψηλότερες διαστάσεις. Οι γενικεύσεις είναι δυνατές για μετρήσεις εκτός από την Ευκλείδεια απόσταση. Ωστόσο, σε αυτές τις περιπτώσεις, ο τριγωνισμός *Delaunay* δεν είναι εγγυημένος ότι υπάρχει ή είναι μοναδικός.

Για ένα σύνολο *P* σημείων στον *d* διαστατικό χώρο, ο τριγωνισμός Delaunay είναι ένας τριγωνισμός έτσι ώστε κανένα σημείο του *P* να μην βρίσκεται μέσα στην περιφέρεια του οποιουδήποτε τοπικού γεωμετρικού περιβλήματος με όρια τα γειτονικά σημεία. Το πρόβλημα της εύρεσης του τριγωνισμού Delaunay ενός συνόλου σημείων στον *n* διαστατικό χώρο μπορεί να μετατραπεί στο πρόβλημα εύρεσης του κυρτού περιβλήματος ενός συνόλου σημείων στον *d + 1* διαστατικό χώρο. Αυτό είναι εφικτό δίνοντας σε κάθε σημείο p μια επιπλέον συντεταγμένη ίση p^2, μετατρέποντας το έτσι σε υπερπαραβολικό λαμβάνοντας την κάτω πλευρά του κυρτού κύτους και γυρνώντας πίσω στον *d* διαστατικό χώρο διαγράφοντας την τελευταία συντεταγμένη. Καθώς το κυρτό κύτος είναι μοναδικό το ίδιο και ο τριγωνισμός του, υποθέτοντας ότι όλες οι όψεις του κυρτού κύτους είναι απλές . Μη απλές όψεις εμφανίζονται μόνο όταν το *n + 2* των αρχικών σημείων βρίσκεται στην ίδια *d - υπερσφαίρα*.

Σχέση με το διάγραμμα Voronoi

Η τριγωνοποίηση *Delaunay* ενός διακριτού σύννεφου σημείων συνόλου *P* στην γενική θέση αντιστοιχεί στο διπλό γράφημα του διαγράμματος *Voronoi*. Οι περιγεγραμμένοι κύκλοι των τριγώνων *Delaunay* είναι οι κορυφές του διαγράμματος *Voronoin*. Στην περίπτωση των 2 διαστάσεων οι κορυφές *Voronoi* συνδέονται μέσω άκρων που προέρχονται από σχέσεις των γειτονικών τρίγωνων *Delaunay* όπως φαίνεται και στην **Εικόνα 2.7**. Εάν δύο τρίγωνα μοιράζονται ένα κοινό άκρο στον τριγωνισμό *Delaunay* τότε οι περιγεγραμμένοι κύκλοι συνδέονται με μία ακμή που ανήκει στο διάγραμμα Voronoi. Ειδικές περιπτώσεις όπου αυτή η σχέση δεν ισχύει ή είναι διφορούμενη περιλαμβάνουν περιπτώσεις όπως:

- Τρία ή περισσότερα γραμμικά σημεία όπου οι περιγεγραμμένοι κύκλοι σχηματίζουν άπειρες ακτίνες.
- Τέσσερα ή περισσότερα σημεία σε έναν τέλειο κύκλο, όπου ο τριγωνισμός είναι ασαφής και όλα τα βαρύκεντρα των τριγώνων κοινά.
- Οι άκρες του διαγράμματος Voronoi που εκτείνονται στο άπειρο δεν καθορίζονται από αυτή τη σχέση στην περίπτωση ενός πεπερασμένο σύνολο *P*. Εάν ο τριγωνισμός *Delaunay* υπολογίζεται χρησιμοποιώντας τον αλγόριθμο Bowyer [9] – Watson [10], τότε πρέπει να αγνοούνται οι τα βαρύκεντρα των τριγώνων που έχουν κοινή κορυφή με το άπειρο τρίγωνο. Οι ακμές του διαγράμματος Voronoi που πηγαίνουν στο άπειρο ξεκινούν από ένα βαρύκεντρο και είναι κάθετες στο κοινό άκρο μεταξύ του διατηρούμενου και του αγνοούμενου τριγώνου.



Εικόνα 2.7: Σχέση τριγώνων Delaunay με το διάγραμμα Voronoi. [ε11]

2.7 Boissonnat

Σε μια πρώιμη εργασία για το θέμα της ανακατασκευής της επιφάνειας ο Boissonnat (1984) πρότεινε δύο διαφορετικές τεχνικές [11]. Η πρώτη του προσέγγιση βασίζεται στην επιφάνεια και τοπικά είναι δισδιάστατη. Ο αλγόριθμος ξεκινά με τον υπολογισμό των k πλησιέστερων γειτόνων (όπου το k είναι παράμετρος εισόδου

στον αλγόριθμο) κάθε δείγματος σημείου. Αυτό επιτυγχάνεται με τη χρήση ενός δέντρου *k-d* (*Bentley and Friedman*) [12]. Μία αυθαίρετα επιλεγμένη άκρη που συνδέει δύο γειτονικά σημεία θεσπίζεται ως τρέχουσα άκρη. Το τοπικό εφαπτομενικό πλάνο της υπολογιζόμενης επιφάνειας στο τρέχον άκρο προσεγγίζεται από ένα επίπεδο ελάχιστων τετραγώνων μέσω των γειτονικών σημείων του τρέχοντος άκρου. Οι άκρες προστίθενται σύμφωνα με έναν κανόνα που επιλέγει τα σημεία από το λίστα των *k* γειτόνων και διατηρεί τοπικά τις ιδιότητες γωνίας Delaunay όταν προβάλλονται ορθογραφικά σημεία στο κατά προσέγγιση εφαπτομενικό επίπεδο. Σε κάθε βήμα προστίθενται τα άκρα και ενημερώνεται το εφαπτόμενο επίπεδο. Η διαδικασία συνεχίζεται έως ότου όλα τα σημεία του δείγματος έχουν προστεθεί στην τριγωνική δομή ακμής. Ο Boissonnat ισχυρίστηκε ότι αυτή η μέθοδος είναι έγκυρη υπό την προϋπόθεση ότι η πυκνότητα των δειγμάτων σε οποιοδήποτε σημείο της επιφάνειας είναι μικρότερη από την ακτίνα καμπυλότητας σε αυτό το σημείο. Το κύριο πρόβλημα με την προσέγγιση του Boissonnat είναι ότι δεν είναι αποτελεσματικό με σύνολα δεδομένων που είναι αραιά και ανομοιόμορφα.

Ο δεύτερος αλγόριθμος που πρότεινε ο Boissonnat στο έγγραφο του ακολουθεί μια ογκομετρική προσέγγιση. Το πρώτο βήμα είναι η κατασκευή ενός τρισδιάστατου τριγωνισμού Delaunay του σύννεφου σημείων. Στη συνέχεια τα τετράεδρα αφαιρούνται από τον τριγωνισμό σύμφωνα με τον ακόλουθο κανόνα όπως αναφέρεται στην δημοσίευσή του: «Το μόνο τετράεδρο που μπορεί να εξαλειφθεί είναι αυτό με ακριβώς ένα πρόσωπο, τρία άκρα και τρία σημεία στο P, ή εκείνα με ακριβώς δύο πρόσωπα, πέντε άκρα και τέσσερα σημεία στο P.» Όπου P είναι το όριο του πολυεδρικού σχήματος που λαμβάνεται με την αφαίρεση της τετραέδρας από τον τριγωνισμό Delaunay. Μετά την αφαίρεση, το P εξάγεται ως μια γραμμική ανακατασκευή της άγνωστης επιφάνειας ως τριγωνικό πλέγμα. Ο Boissonnat αποκαλεί τη διαδικασία αφαίρεσης «γλυπτική». Το μεγαλύτερο μειονέκτημα στη γλυπτική προσέγγιση είναι ότι είναι πιθανό ορισμένα σημεία του δείγματος να παραμένουν εντός του P κατά τον τερματισμό του αλγορίθμου.

2.8 Αλγόριθμος μηδενικού συνόλου Hoppe

Ο Hoppe αντιμετωπίζει το πρόβλημα της ανακατασκευής μιας επιφάνειας από μη οργανωμένα σημεία χωρίς χρήση οποιαδήποτε άλλη γνώση της επιφάνειας του δείγματος όπως γειτονικές πληροφορίες [13]. Δεν δίνουν καμία εγγύηση σύγκλισης ή ορθότητας για τον αλγόριθμό τους αλλά ισχυρίζονται ότι λειτουργεί καλά στην πράξη. Η μέθοδος τους ονομάζεται αλγόριθμος μηδενικού συνόλου επειδή προσεγγίζει μια επιφάνεια που σχηματίζεται από το σύνολο σημείων που έχουν απόσταση μηδέν από την επιφάνεια.

1. <u>Απαιτούμενες πληροφορίες δειγματοληψίας</u>

Όπως ειπώθηκε αν και ο αλγόριθμος δεν απαιτεί πληροφορίες σχετικά με τα σημεία δείγματος εκτός από τις συντεταγμένες τους, παρόλα αυτά απαιτεί γνώση της διαδικασίας δειγματοληψίας. Χρησιμοποιούνται δύο δειγματοληπτικοί παράμετροι:

- Το μέγεθος θορύβου *δ*: το μέγιστο σφάλμα δειγματοληψίας που για παράδειγμα είναι η ακρίβεια ενός σαρωτή λέιζερ.
- Την πυκνότητα δειγματοληψίας *ρ*: η ακτίνα μιας σφαίρας έτσι ώστε οποιαδήποτε σφαίρα με ακτίνα *ρ* και κέντρο *O* στην άγνωστη επιφάνεια M

περιέχει τουλάχιστον ένα σημείο δείγματος. Διαισθητικά, το *ρ* αντιπροσωπεύει την ακτίνα της μέγιστης οπής στο σύνολο δεδομένων του δείγματος.

2. Υπολογισμός εφαπτομενικού επιπέδου

Το πρώτο βήμα του αλγορίθμου είναι να εκτιμηθεί ένα εφαπτόμενο επίπεδο σε κάθε σημείο δείγματος. Τα εφαπτόμενα επίπεδα προσεγγίζουν την εφαπτομένη στην άγνωστη επιφάνεια *M* σε κάθε ένα από τα σημεία του δείγματος. Η γειτονιά *k* ορίζεται ως τα *k* πλησιέστερα σημεία που περιβάλλουν ένα σημείο δείγματος, όπου το *k* είναι μια παράμετρος που καθορίζεται από το χρήστη. Τα εφαπτόμενα επίπεδα υπολογίζονται τοποθετώντας ένα τετράγωνο επίπεδο διαμέσου σημείων στην *k* γειτονιά κάθε σημείου δείγματος. Τα εφαπτόμενα επίπεδα καθορίζονται από το κεντρικό τους σημείο *O* και από τον κανονικό διάνυσμα *n*.

3. Προσανατολισμός εφαπτομενικού πεδίου

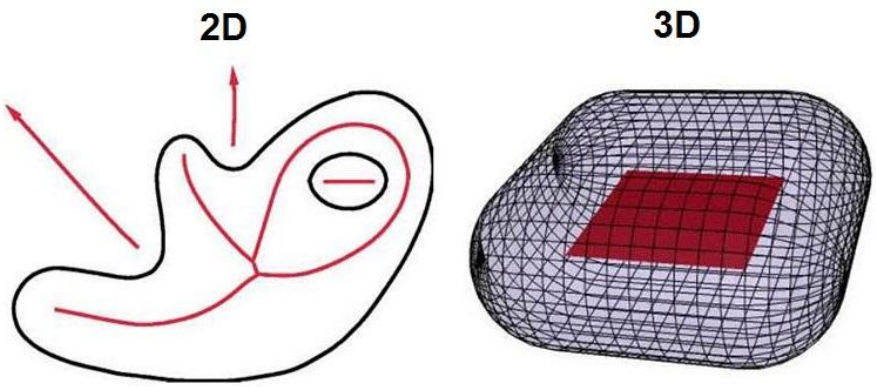
Στην επόμενη φάση του αλγορίθμου βρίσκεται ο προσανατολισμός για τα εφαπτόμενα επίπεδα. Όλοι οι κανονικοί φορείς του εφαπτόμενου επιπέδου πρέπει να δείχνουν μακριά από το αντικείμενο. Επομένως, είναι απαραίτητο να γίνει αντιστροφή των διανυσμάτων των εφαπτόμενων επιπέδων που δείχνουν στην λάθος κατεύθυνση. Το πρώτο βήμα για τον προσανατολισμό των εφαπτόμενων επιπέδων είναι ο υπολογισμός του ελαχίστου γεννητικού δέντρου των σημείων του γραφήματος που συνδέει τα εφαπτόμενα κέντρα O_i. Το ελάχιστο γεννητικό δέντρο σημείων «*MST*» είναι ένα ελάχιστο δέντρο έκτασης ενός συνδεδεμένου γραφήματος σημείων ώστε το συνολικό μήκος των γραμμών που συνδέουν όλα τα σημεία να είναι το ελάχιστο. Οι συγγραφείς διαπίστωσαν ότι το *MST* είναι αρκετά πυκνό στα άκρα για την εύρεση του προσανατολισμού του επιπέδου για να λειτουργεί σωστά. Για την επίλυση αυτού του προβλήματος το *MST* εμπλουτίζεται προσθέτοντας αντίστοιχα άκρα στα κέντρα του εφαπτόμενου επιπέδου που βρίσκονται στην ίδια γειτονιά k. Το γράφημα που προκύπτει ονομάζεται γράφημα *Riemannian*. Η ανατροπή των επιπέδων λαμβάνει χώρα κατά τη διάρκεια της διασταύρωσης του γραφήματος *Riemannian*. Επομένως, το γράφημα πρέπει να σαρώνεται κατά μήκος ενός συνόλου διαδρομών με τις άκρες μεταξύ των εφαπτόμενων επιπέδων που είναι παράλληλες σε αυτά. Το αρχικό επίπεδο ρυθμίζεται ώστε να δείχνει προς την κατεύθυνση του άξονα *+z* μακριά από το αντικείμενο. Τέλος, το *MST* διασχίζεται σε βάθος και κατά τη διέλευση από κάθε εφαπτομενικό επίπεδο ρυθμίζεται έτσι ώστε να δείχνει προς μια κατεύθυνση που είναι σύμφωνη με το αρχικό εφαπτόμενο επίπεδο.

4. Υπολογισμός αποστάσεων

Σε επόμενο στάδιο του αλγόριθμου Hoppe, είναι απαραίτητο να βρεθεί η απόσταση *d* από το πλησιέστερο σημείο *p* της άγνωστης επιφάνειας *M*. Δεδομένου ότι η επιφάνεια *M* είναι άγνωστη, η λύση που βρήκαν οι συγγραφείς για να υπολογίσουν το *d* είναι να χρησιμοποιήσουν το κέντρο του εφαπτόμενου επίπεδου να είναι το πλησιέστερο στο *p* ως προσέγγιση της απόστασης από την άγνωστη επιφάνεια *M*. Η απόσταση είναι θετική εάν το *p* είναι μπροστά από το εφαπτόμενο επίπεδο, μηδέν εάν το *p* βρίσκεται στο επίπεδο εφαπτομένης και αρνητικό αν το *p* είναι πίσω από το επίπεδο. Έστω το *D* να είναι η απόσταση μεταξύ του *p* και του πλησιέστερου εφαπτομενικού κέντρου, τότε η απόσταση d θεωρείται απροσδιόριστη εάν .

2.9 Αλγόριθμος Crust

Ο αλγόριθμος Crust αξιοποιεί την έννοια του μεσαίου άξονα για να περικλείσει τον τρισδιάστατο τριγωνισμό *Delaunay* των δειγμάτων δεδομένων της άγνωστης επιφάνειας [14]. Ο διάμεσος άξονας ορίζεται ως ο γεωμετρικός τόπος των σημείων που έχουν περισσότερα από ένα πλησιέστερα σημεία στην καμπύλη για τις δύο διαστάσεις ή στην επιφάνεια για τρεις διαστάσεις, ένα παράδειγμα φαίνεται στην **Εικόνα 2.8**.

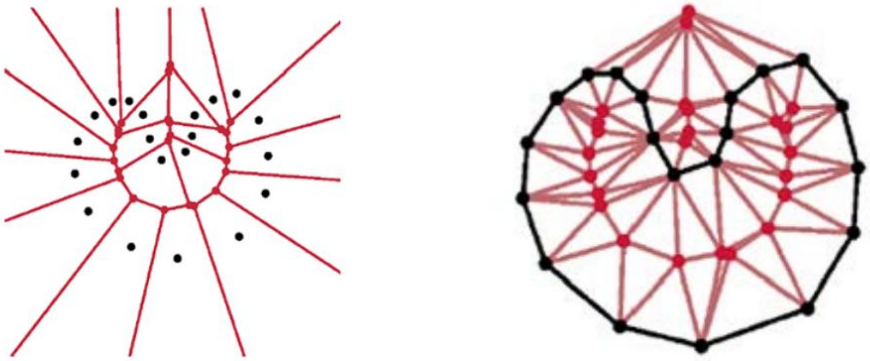


Εικόνα 2.8: Διάμεσος άξονας και επίπεδο. [ε12]

Δισδιάστατος αλγόριθμος

1. Υπολογισμός του διαγράμματος Voronoi των σημείων δείγματος στις δύο διαστάσεις.
2. Κατασκευή του τριγωνισμού *Delaunay* των σημείων δείγματος και των κορυφών *Voronoi* στις δύο διαστάσεις.
3. Η *2D crust* (ανακατασκευασμένη καμπύλη) σχηματίζεται εξάγοντας όλες τις άκρες που συνδέουν δύο σημεία δείγματος από τον τριγωνισμό που πραγματοποιήθηκε στο Βήμα 2.

Σε δύο διαστάσεις οι κορυφές *Voronoi* προσεγγίζουν τον μεσαίο άξονα όπως φαίνεται και στην **Εικόνα 2.9**. Η προσθήκη των κορυφών *Voronoi* τείνει σπάσει τα άκρα που συνδέουν μη παρακείμενα σημεία δείγματος και αυτό θα ισχύει πάντα για ένα σύνολο δεδομένων με επαρκή δειγματοληψία. Οι συγγραφείς αποκαλούν τη διαδικασία αφαίρεσης ακμών μεταξύ μη γειτονικών σημείων δειγματοληψίας ως φιλτράρισμα Voronoi.



Εικόνα 2.9: Σχέση διαγράμματος Voronoi με τον μεσαίο άξονα. [ε12]

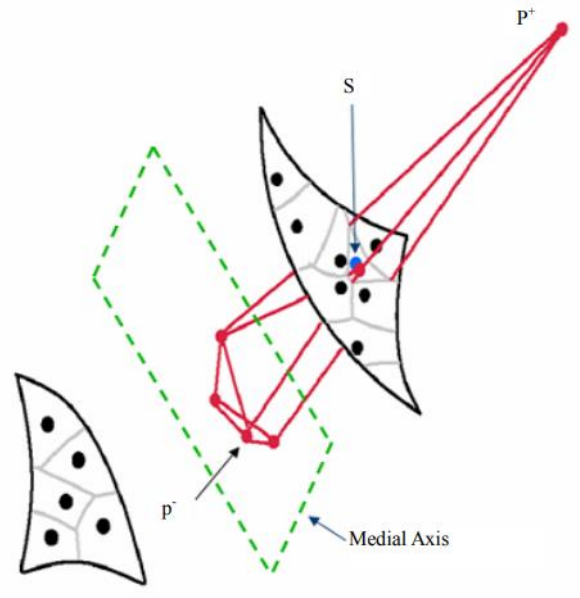
Ο αλγόριθμος *2D Crust* μπορεί να επεκταθεί σε 3D με ορισμένες τροποποιήσεις. Η κύρια δυσκολία στην επέκταση του είναι ότι οι κορυφές *Voronoi* στις τρεις διαστάσεις δεν βρίσκονται απαραίτητα κοντά στον μεσαίο άξονα. Ευτυχώς, παρόλο που οι κορυφές *Voronoi* δεν είναι εγγυημένες ότι βρίσκονται κοντά στον μεσαίο άξονα οι περισσότερες είναι. Στο βήμα φιλτραρίσματος *3D Voronoi* αντί να χρησιμοποιούνται όλες οι κορυφές Voronoi οι συγγραφείς χρησιμοποίησαν τις δύο κορυφές *Voronoi* που είναι πιο απομακρυσμένες από κάθε σημείο δείγματος σε αντίθετες πλευρές της άγνωστης επιφάνειας. Καλούν αυτές τις κορυφές τους πόλους του σημείου δείγματος. Οι πόλοι του νέφους σημείων λαμβάνονται σε δύο στάδια:

1. Βρίσκεται κορυφή Voronoi που βρίσκεται πιο μακριά από το σημείο *s* δείγματος. Αυτή η κορυφή είναι ο θετικός πόλος, που υποδηλώνεται *p+*.
2. Βρίσκεται η κορυφή Voronoi πιο μακριά από *s* έτσι ώστε το εσωτερικό γινόμενο των *sp+* και *sp-* να είναι αρνητικό. Αυτό διασφαλίζει ότι οι πόλοι βρίσκονται σε αντίθετες πλευρές της επιφάνειας.

Τρισδιάστατος αλγόριθμος

Επομένως για σύννεφο σημείων στις τρεις διαστάσεις ο αλγόριθμος *3D Crust* μπορεί να περιγραφεί ως εξής:

1. Υπολογισμός του διαγράμματος Voronoi των σημείων δείγματος σε τρεις διαστάσεις.
2. Κατασκευή του τριγωνισμού Delaunay των σημείων δείγματος και εύρεση των πόλων σημείων του δείγματος (*p+* και *p-*) στις τρεις διαστάσεις.
3. Η 3D crust (ανακατασκευασμένη επιφάνεια) σχηματίζεται εξάγοντας όλα τα τρίγωνα που συνδέουν τρία σημεία του δείγματος από τον τριγωνισμό που διεξήχθη στο δεύτερο βήμα όπως το παράδειγμα στην **Εικόνα 2.10**.



Εικόνα 2.10: Αλγόριθμος Crust στις τρεις διαστάσεις. [ε12]

3.ΑΡΧΗ ΛΕΙΤΟΥΡΓΙΑΣ

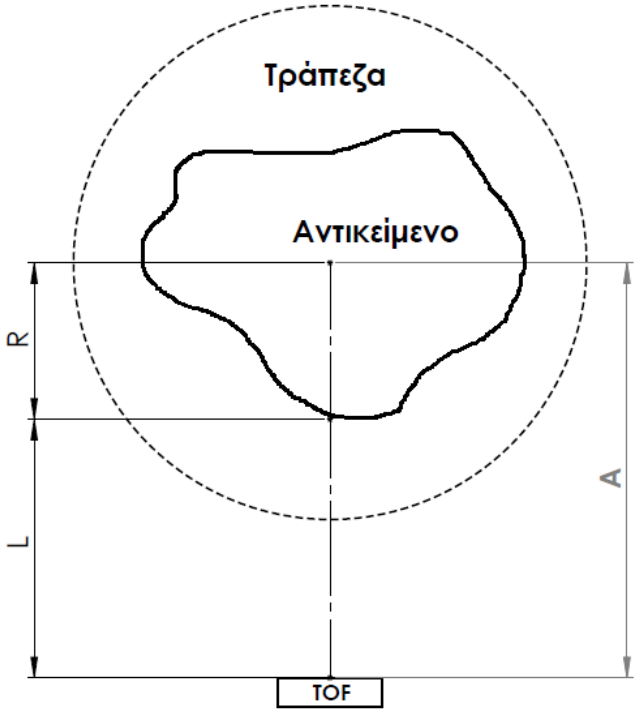
3.1 Λειτουργία σαρωτή

Όπως ειπώθηκε σε προηγούμενο κεφάλαιο, στα πλαίσια της παρούσας πτυχιακής θα υλοποιηθεί τρισδιάστατος σαρωτής αισθητήρα TOF. Ο συγκεκριμένος αισθητήρας διαθέτει μία πηγή φωτός και μετρώντας την χρονική διαφορά επιστροφής η την διαφορά φάσης των επιστρεφόμενων σημάτων φωτός σε σχέση με τα εκπεμπόμενα μπορεί να υπολογίσει την απόσταση μεταξύ αυτού και ενός σημείου της επιφάνειας που προσπίπτει το φως. Με βάση αυτή την αρχή είναι δυνατόν να κατασκευαστεί διάταξη η οποία θα μετράει και θα υπολογίζει τις συντεταγμένες σημείων επάνω στην επιφάνεια ενός αντικειμένου.

Σάρωση και μέτρηση επιφάνειας

Προκειμένου να σαρωθεί και να μετρηθεί όλη η επιφάνεια ενός αντικειμένου η διάταξη θα πρέπει να διαθέτει μία περιστρεφόμενη τράπεζα όπου θα τοποθετείται το αντικείμενο, ενώ ο αισθητήρα να έχει την δυνατότητα κίνησης κατά το ύψος στον άξονα Z. Η περιστρεφόμενη τράπεζα θα οδηγείται από βηματικό κινητήρα ελεγχόμενης γωνίας καθώς και ο αισθητήρας θα μετακινείται με συνδυασμό κοχλία περικοχλίου, με τον κοχλία συνδεδεμένο και επάνω σε βηματικό ηλεκτροκινητήρα. Οι μετρήσεις των σημείων επάνω στην επιφάνεια του αντικειμένου θα πραγματοποιούνται δειγματοληπτικά, η περιστρεφόμενη τράπεζα θα διαγράφει κάθε φορά μία μικρή γωνία και θα σταματάει έτσι ώστε ο αισθητήρας να καταγράψει την μέτρηση. Αφού πραγματοποιηθεί η μέτρηση η τράπεζα θα περιστρέφεται πάλι κατά μία μικρή ορισμένη γωνία και θα ξανά λαμβάνεται μέτρηση. Όταν πραγματοποιηθεί

ένα πλήρης κύκλος τότε ο αισθητήρας μέτρησης απόστασης θα ανυψώνεται ελάχιστα σε γνωστά διαστήματα και η διαδικασία θα επαναλαμβάνεται έως ότου σαρωθεί το αντικείμενο περιμετρικά και σε όλο το ύψος του. Επομένως στην πρώτη φάση γίνεται μία δειγματοληψία σημείων της επιφάνειας εκφρασμένες σε πολικές συντεταγμένες από τον κέντρο περιστροφής της τράπεζας όπως φαίνεται και στην παρακάτω **Εικόνα 3.1**.



Εικόνα 3.1: Αποστάσεις υπολογισμού ακτίνας της διάταξης.

Ο υπολογισμός της ακτίνας *R* του σημείου από το κέντρο περιστροφής της τράπεζας δίνεται από τον τύπο:

(3.1.1)

Όπου:
L = Η μετρούμενη απόσταση που διαβάζει ο αισθητήρας.
A = Η γνωστή απόσταση του αισθητήρα από το κέντρο περιστροφής της τράπεζας.

Διευκρινίζεται ότι κατά την πρώτη μέτρηση θέτεται $\theta = 0$ και $Z=0$. Το θ κάθε φορά αυξάνει κατά γνωστή γωνία θ_B που αντιστοιχεί στο βήμα της γωνίας και ξαναμηδενίζεται σε μία πλήρη περιστροφή αφού το Z αυξηθεί και αυτό κατά γνωστό ύψος Z_B. Σε κάθε νέο βήμα προσθέτεται το βήμα στην προηγούμενη συνολική τιμή του κάθε μεγέθους αντίστοιχα:

(3.1.2)

(3.1.3)

Αυτό έχει ως αποτέλεσμα να δημιουργηθεί μία ακολουθία (R, θ, Z) όπου είναι καταγεγραμμένα όλα τα σημεία της επιφάνειας που σαρώθηκε σε κυλινδρικές συντεταγμένες.

Αποθήκευση συντεταγμένων

Εκτός από την μέτρηση της απόστασης σημαντικό για να δημιουργηθεί το νέφος σημείων της επιφάνειας, είναι η αποθήκευση των καταγεγραμμένων συντεταγμένων. Η συσκευή έχει την δυνατότητα να αποθηκεύσει τις μετρήσεις του αισθητήρα σε μία κάρτα μνήμης ή και να τις μεταδώσει σε πραγματικό χρόνο μέσω θύρας *USB* στον υπολογιστή όπου τα παραλαμβάνει το λογισμικό σε Python που έχει υλοποιηθεί και τα αποθηκεύει στην *ROM* του υπολογιστή. Ο μικροελεγκτής διαβάζει τις μετρήσεις από τον αισθητήρα *TOF* μέσω του πρωτοκόλλου επικοινωνίας *I2C*. Η εγγραφή στην κάρτα μνήμης γίνεται με την χρήση του πρωτοκόλλου *SPI* και στην περίπτωση της επικοινωνίας με τον υπολογιστή ένα κύκλωμα *USB to Serial* επάνω στον ίδιο τον μικροελεγκτή επιτρέπει την διαχείριση και αποστολή των δεδομένων με εντολές *UART Serial*.

Δημιουργία νέφους

Το λογισμικό το οποίο έχει δημιουργηθεί γι αυτή την πτυχιακή έχει σκοπό τον έλεγχο διαδικασίας του σαρωτή και την επεξεργασία των μετρήσεων για να παραχθεί η τελική μορφή του νέφους σε καρτεσιανές συντεταγμένες *XYZ* και εν τέλει να παραχθεί το τελικό αρχείο *STL* το οποίο είναι αναγνώσιμο από λογισμικά *CAD*. Για την ακρίβεια ο ελεγκτής καταγράφει στην κάρτα μνήμης η στέλνει μέσω *USB* μόνο της τιμές μέτρησης απόστασης του αισθητήρα *TOF*. Παρόλα αυτά είναι γνωστός ο αριθμός των σημείων λήψης σε μία ολόκληρη περιστροφή όπως επίσης και το ύψος ανύψωσης κάθε φορά του αισθητήρα. Επομένως το λογισμικό μπορεί αρχικά να εξάγει τις κυλινδρικές συντεταγμένες (R, θ, Z) και με εφαρμογή μαθηματικών τύπων να τις μετατρέψει σε καρτεσιανές συντεταγμένες (X, Y, Z) που δίνονται παρακάτω:

(3.1.4)

(3.1.5)

Δημιουργία STL

Αφού αποθηκευτούν οι καρτεσιανές συντεταγμένες (X, Y, Z) σε ένα αρχείο κειμένου, αυτό χρησιμοποιείται μετέπειτα ως είσοδο για την παραγωγή του αρχείου *STL* είτε στον κώδικα *MATLAB* που είναι ενταγμένος στο λογισμικό της *Python* του σαρωτή, είτε σε άλλα προγράμματα επεξεργασίας νέφους σημείων και υπολογισμού επιφανειών όπως για παράδειγμα το *MeshLab*.

3.2 Λειτουργία βηματικών κινητήρων

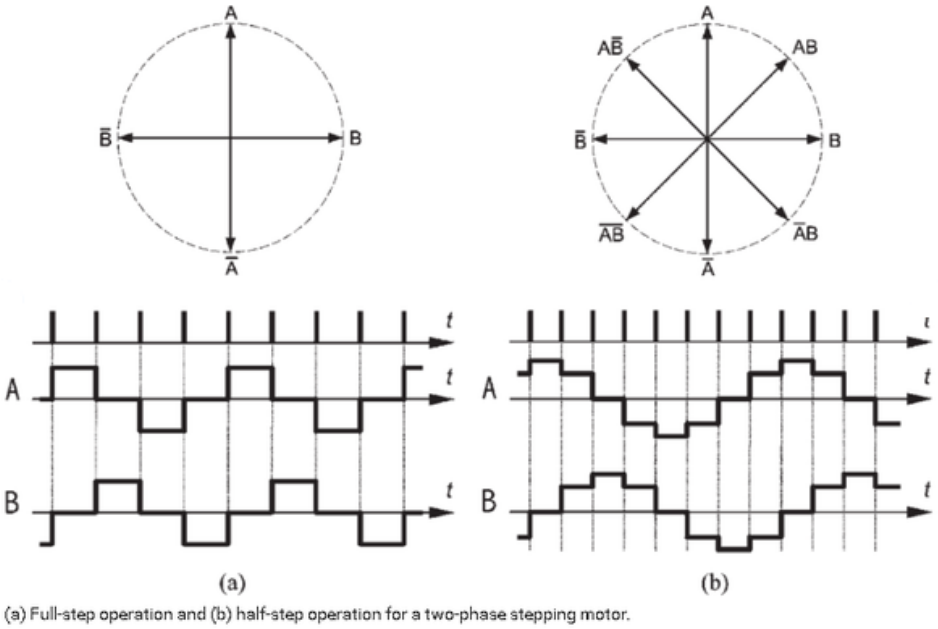
Οι βηματικοί κινητήρες ή αλλιώς κινητήρες διακριτών βημάτων περιστροφής «stepper motors» είναι ηλεκτρικοί κινητήρες επαγωγής με ελεγχόμενη γωνία περιστροφής συστήματος ανοιχτού βρόγχου του άξονα που επιτυγχάνεται με κατάλληλο ηλεκτρονικό κύκλωμα οδήγησης.

Κύρια μέρη του κινητήρα είναι ο στάτης και ο δρομέας. Ο στάτης φέρει τα τυλίγματα που περιελίσσονται στον πυρήνα των πόλων σε ζεύγη. Σε κάθε ζεύγος τα

τυλίγματα του κάθε πόλου είναι σε σειρά και περιελιγμένα έτσι ώστε να δημιουργείται πολικότητα μεταξύ τους [15]. Ο δρομέας συνήθως αποτελείται από μόνιμους μαγνήτες ή από οδοντωτό τροχό μαγνητικού υλικού στην περίπτωση των βηματικών κινητήρων μεταβλητής μαγνητικής αντίδρασης. Το μαγνητικό υλικό του στάτη και του δρομέα συχνά είναι συμπαγές αλλά προτιμάται να αποτελείται από ελάσματα κράματος χάλυβα πυριτίου για την μείωση των απωλειών ενέργειας λόγω δινορευμάτων. Τα υλικά αυτά παρουσιάζουν μεγάλη μαγνητική διαπερατότητα και έτσι με οποιαδήποτε εφαρμογή τάσης δημιουργείται υψηλή μαγνητική ροή. Επομένως ο αριθμός των γωνιακών βημάτων του κινητήρα είναι ανάλογος του αριθμού των δοντιών του δρομέα.

Η λειτουργία τους βασίζεται στις βασικές αρχές του μαγνητισμού. Οι ομώνυμοι μαγνητικοί πόλοι απωθούνται και οι ετερώνυμοι έλκονται. Η πολικότητα των τυλιγμάτων του στάτη καθορίζεται από την φορά του ρεύματος που τα διαρρέει, ο δρομέας αντίστοιχα υφίσταται ροπή μέχρι να ισορροπήσει σε γωνία όπου οι ετερώνυμοι πόλοι μεταξύ δρομέα και στάτη έρθουν σε ευθυγράμμιση. Συνήθως η ανάλυση περιστροφής ολόκληρων βημάτων είναι τυπικά 200 ή 400 βήματα.

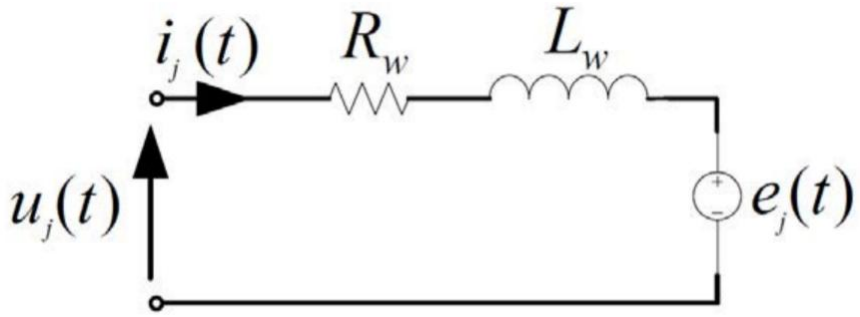
Κάθε ηλεκτρονικό κύκλωμα οδήγησης βηματικού κινητήρα δέχεται ως είσοδο παλμούς και εφαρμόζοντας τάση στα τυλίγματα του στάτη περιστρέφει τον άξονα κατά βήμα ανά παλμό. Μία πολύ σημαντική λειτουργία είναι αυτή της υποδιαίρεσης των βημάτων περιστροφής. Το κύκλωμα οδήγησης εφαρμόζει ενδιάμεσες τιμές τάσης στα τυλίγματα έτσι ώστε το σημείο ισορροπίας του αντίθετου πόλου το δρομέα να μην είναι ευθυγραμμισμένο με τον πόλο του τυλίγματος αλλά σε ενδιάμεσες θέσεις. Αυτό συμβαίνει καθώς ο πόλος του δρομέα δέχεται έλξη και από το προηγούμενο και από το επόμενο τύλιγμα και το σημείο ισορροπίας βρίσκεται σε ενδιάμεση θέση και πιο κοντά στον πόλο με την μεγαλύτερη δύναμη έλξης όπως φαίνεται και στην **Εικόνα 3.2**. Η κίνηση σε υποδιαιρέσεις βημάτων αυξάνει την ελεγχόμενη ακρίβεια αλλά και προσφέρει ομαλή περιστροφή του άξονα με λιγότερες δονήσεις και σημαντική μείωση του θορύβου λειτουργίας.



(a) Full-step operation and (b) half-step operation for a two-phase stepping motor.

Εικόνα 3.2: Λειτουργία υποδιαίρεσης βημάτων. [ε13]

Κάθε ηλεκτρική φάση ή τύλιγμα του βηματικού κινητήρα για χαμηλές συχνότητες και ρεύματα μπορεί να μοντελοποιηθεί ως ένα *RL* κύκλωμα συν την ηλεκτρεγερτική δύναμη όπως φαίνεται και στην **Εικόνα 3.3**.



Εικόνα 3.3: Απλό ισοδύναμο κύκλωμα τυλίγματος βηματικού κινητήρα. [ε14]

—— (3.2.1)

Όπου:
V(t) = εφαρμοστέα τάση
e(t) = ηλεκτρεγερτική δύναμη
R = αντίσταση τυλίγματος
I = ρεύμα τυλίγματος
L = επαγωγή τυλίγματος

Η ηλεκτρεγερτική δύναμη κάθε τυλίγματος περιγράφεται από την παρακάτω σχέση:

(3.2.2)

(3.2.3)

Όπου:
K_m = σταθερά κινητήρα
p = αριθμός ζευγών πόλων
ω_n = γωνιακή ταχύτητα
θ_m = γωνία

Εφαρμόζοντας τον μετασχηματισμό Laplace στην πρώτη σχέση καταλήγουμε:

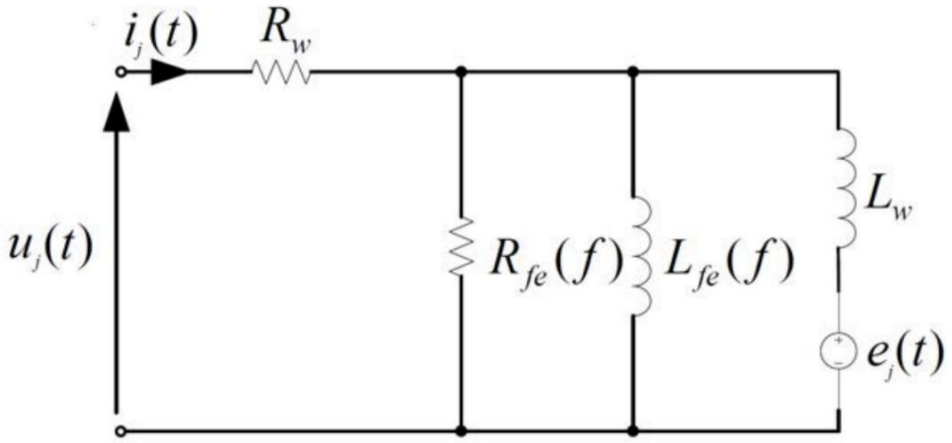
—— (3.2.4)

και

(3.2.5)

Όπου:
Z = σύνθετη αντίσταση

Παρόλα αυτά σε υψηλότερες συχνότητες δεν είναι αρκετό το μοντέλο ισοδύναμου κυκλώματος *RL*. Αυτό γιατί τα σιδηρομαγνητικά υλικά έχουν απώλειες που εξαρτώνται άμεσα από την συχνότητα. Το ισοδύναμο κύκλωμα υψηλών συχνοτήτων παρουσιάζεται στην παρακάτω **Εικόνα 3.4**:



Εικόνα 3.4: Ισοδύναμο κύκλωμα υψηλών συχνοτήτων τυλίγματος βηματικού κινητήρα. [ε14]

Η σύνθετη αντίσταση για το ισοδύναμο κύκλωμα υψηλής συχνότητας υπολογίζεται πλέον:

——— (3.2.6)

Όπου *Leq* ισοδύναμη επαγωγή από την παραλληλία των *Lfe* και *Lw*:

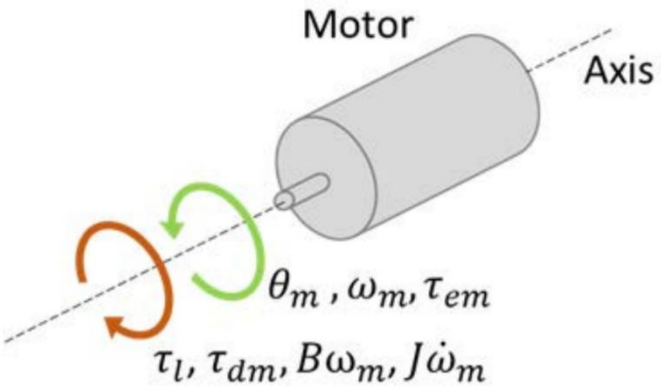
——— (3.2.7)

και

— (3.2.8)

Όπου Rfe και Lfe απώλειας λόγω αντίστασης και επαγωγής του σιδηρομαγνητικού πυρήνα.

Για την μοντελοποίηση του μηχανικού μέρους του βηματικού κινητήρα θα χρησιμοποιηθεί ο δεύτερος νόμος του Νεύτωνα θεωρώντας τον δρομέα και το συνδεδεμένο φορτίο στον άξονα ως ένα συμπαγές σώμα όπως στην **Εικόνα 3.5**.



Εικόνα 3.5: Μηχανικό μοντέλο βηματικού κινητήρα. [ε14]

Από τον δεύτερο νόμο του Νεύτωνα για περιστροφική κίνηση ισχύει ότι [δ14]:

— (3.2.9)

Υπολογίζεται ότι:

— (3.2.10)

και

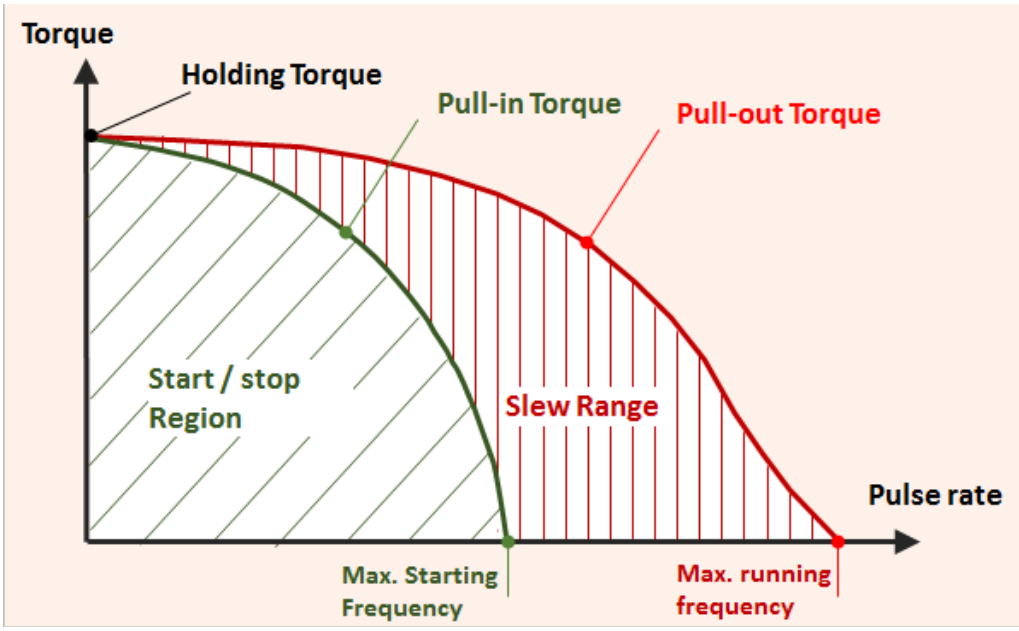
(3.2.11)

και

(3.2.12)

Όπου:
τ_{mot} = ροπή κινητήρα
J = ροπή αδρανείας
B = συντελεστής τριβής ιξώδους
τ_{dm} = Ροπή συγκράτησης
T_{dm} = πλάτος ροπής συγκράτησης
φ = μετατόπιση φάσης
τ_l = ροπή φορτίου

Η χαρακτηριστική ροπής-ταχύτητας περιστροφής παρουσιάζεται στην παρακάτω **Εικόνα 3.6**.



Εικόνα 3.6: Χαρακτηριστική καμπύλη ροπής-ταχύτητας βηματικού κινητήρα. [ε15]

Παρατηρείται ότι με αύξηση των στροφών του κινητήρα η ροπή του κινητήρα μειώνεται δραματικά. Επίσης διακρίνονται δύο περιοχές λειτουργίας:

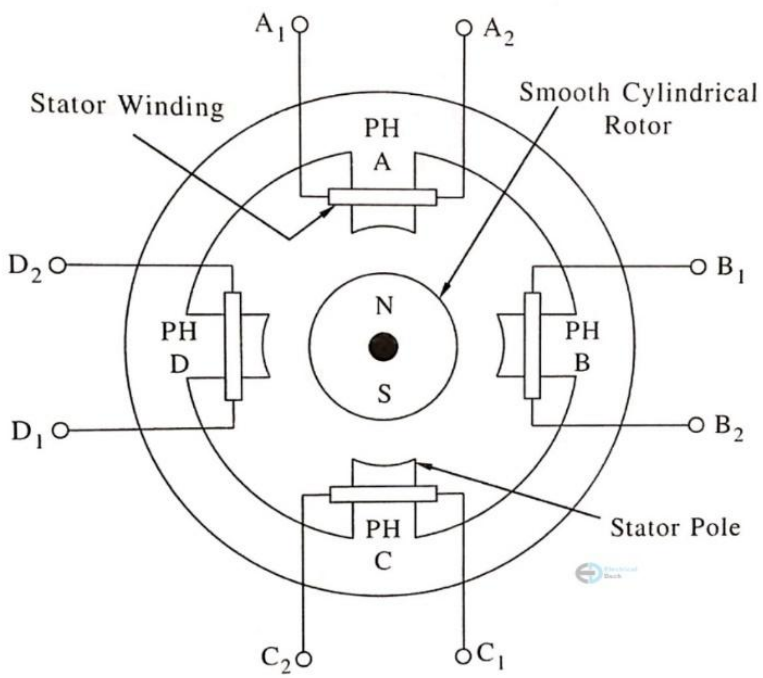
- Κατάσταση εκκίνησης-ακινητοποίησης (*start - stop*)
- Κατάσταση συνεχούς περιστροφής (*slew range*)

Στην περιοχή *start - stop* ο βηματικός κινητήρας μπορεί να εκκινήσει ακαριαία την περιστροφή του χωρίς να χρειάζεται να υπάρξει επιτάχυνση στις επιθυμητές στροφές. Αντίθετα στην περιοχή *slew range* ο κινητήρας μπορεί να οδηγεί το φορτίο στις ανάλογες στροφές μόνο όταν βρίσκεται ήδη σε κίνηση ή έχει θα επιταχυνθεί στις επιθυμητές στροφές. Και για τις δύο περιπτώσεις αντίστοιχα πέρα από τις καμπύλες *Pull - in Torque* και *Pull-out Torque* ο κινητήρας αποσυγχρονίζεται αν δεν υφίστανται οι κατάλληλες συνθήκες. Είναι σημαντικό λοιπόν να ορίζονται ακριβώς οι απαιτήσεις της εφαρμογής και να επιλέγεται βηματικός κινητήρας που πληροί τις απαιτήσεις οδήγησης καθώς πρόκειται για ένα σύστημα ελέγχου ανοιχτού βρόγχου και σε περίπτωση σφάλματος ή αστοχίας του ελέγχου θέσης-ταχύτητας το ηλεκτρονικού κύκλωμα δεν το γνωρίζει ώστε να κάνει τις απαραίτητες διορθώσεις.

Είδη βηματικών κινητήρων

Υπάρχουν 3 είδη βηματικών κινητήρων ανάλογα με την κατασκευή του δρομέα:

1. Μόνιμου Μαγνήτη

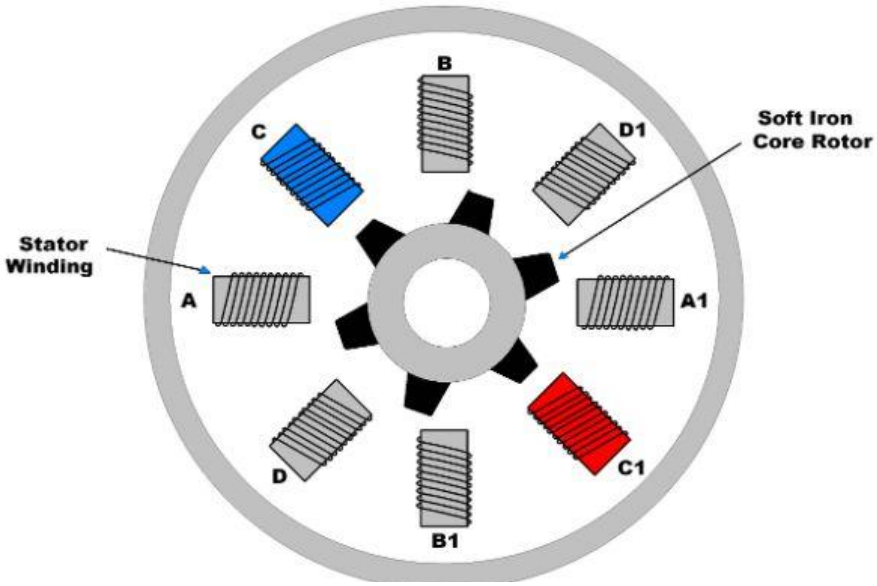


Εικόνα 3.7: Σχηματικό βηματικού κινητήρα μόνιμου μαγνήτη. [ε16]

Στους βηματικούς κινητήρες μόνιμου μαγνήτη όπως λέει και το όνομα τους, ο δρομέας κατασκευάζεται από μόνιμους μαγνήτες σε εναλλασσόμενη συστοιχία πολικότητας περιμετρικά όπως στην **Εικόνα 3.7**. Αλλάζοντας την φορά του μαγνητικού πεδίου του στάτη οι μόνιμοι μαγνήτες έλκονται από την αντίθετη πολικότητα του αντίστοιχου τυλίγματος του στάτη και ο άξονας ισορροπεί σε συγκεκριμένη γωνιά πραγματοποιώντας ένα βήμα από το προηγούμενο σημείο ισορροπίας. Η συχνότητα εναλλαγής του μαγνητικού πεδίου του στάτη καθορίζει και την ταχύτητα περιστροφής του άξονα.

Να σημειωθεί ότι οι βηματικοί κινητήρες αυτού του είδους έχουν μεγάλες γωνίες βηματισμού είναι χαμηλής ταχύτητας και ροπής αλλά χαμηλού κόστους. Βρίσκουν εφαρμογή κυρίως σε εκτυπωτές και ηλεκτρονικές συσκευές που απαιτείται έλεγχος κίνησης.

2. Μεταβλητής μαγνητικής αντίδρασης



Εικόνα 3.8: Σχηματικό βηματικού κινητήρα μεταβλητής μαγνητικής αντίδρασης. [ε17]

Πρόκειται για τους πιο διαδεδομένους βηματικούς κινητήρες καθώς προσφέρουν μεγαλύτερη γωνιακή ανάλυση και ροπή. Ο δρομέας κατασκευάζεται από συμπαγές ή ελασματοποιημένο μαλακό μαγνητικό υλικό με υψηλή μαγνητική διαπερατότητα. Ο δρομέας περιστρέφεται μέσω της διέγερσης τους κατάλληλου πόλου του στάτη έτσι ώστε κάποιος ζεύγος αντίστροφης πολικότητας μεταξύ του δρομέα και του στάτη να ευθυγραμμιστεί **Εικόνα 3.8**. Χρησιμοποιείται συχνά σε βιομηχανικές εφαρμογές ή σε μηχανήματα CNC χαμηλής ισχύος. Αναφορικά σε περιπτώσεις ελέγχου μεγάλης ισχύος χρησιμοποιούνται σερβοκινητήρες κλειστού ελέγχου σε αντικατάσταση των βηματικών κινητήρων.

3. Υβριδικοί

Οι υβριδικοί βηματικοί κινητήρες συνδυάζουν τα χαρακτηριστικά των δύο προηγούμενων. Ο δρομέας αποτελείται από μόνιμους μαγνήτες όπου στα άκρα τους προσαρμόζονται δόντια από μαγνητικού υλικού όμοια με του κινητήρα μεταβλητής μαγνητικής αντίδρασης. Λόγω του μόνιμου μαγνήτη μαγνητίζονται και αυτά και αποκτούν πολικότητα. Συνοψίζοντας από τα παραπάνω:

Πλεονεκτήματα

- Εξαιρετική απόκριση στην εκκίνηση και ακινητοποίηση χωρίς να απαιτείται μηχανικό φρένο
- Ροπή συγκράτησης σε ακινησία
- Μεγάλη ακρίβεια ελέγχου θέσης

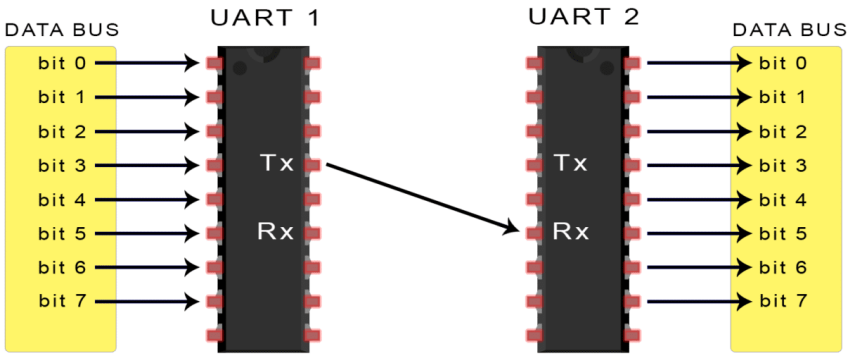
- Επιτυγχάνει χαμηλές ταχύτητες περιστροφής με μεγάλη ροπή
- Αντοχή στον χρόνο καθώς δεν διαθέτει κινούμενες ηλεκτρικές επαφές
- Μικρό κόστος και μεγάλη διαθεσιμότητα μοντέλων

Μειονεκτήματα

- Μικρή ροπή σε υψηλές ταχύτητες
- Θορυβώδεις κατά την λειτουργία τους
- Έχει κακή αναλογία μεγέθους κατασκευής και δύναμης οδήγηση

3.3 UART

Ένας καθολικός ασύγχρονος πομπός / δέκτης «*UART*» είναι ένα μπλοκ ολοκληρωμένου κυκλώματος που είναι υπεύθυνο για την λειτουργία της ασύγχρονης σειριακής επικοινωνίας. Ουσιαστικά, το *UART* λειτουργεί ως μεσάζοντας μεταξύ παράλληλων και σειριακών διεπαφών. Στο ένα άκρο του *UART* υπάρχει ένας δίαυλος με οκτώ περίπου γραμμές δεδομένων και από την άλλη μεριά είναι δύο σειριακά καλώδια *RX* και *TX* όπως φαίνεται στην **Εικόνα 3.9**. Ένα από τα καλύτερα χαρακτηριστικά για το *UART* είναι ότι χρησιμοποιεί μόνο δύο καλώδια για τη μετάδοση δεδομένων μεταξύ συσκευών [δ15].

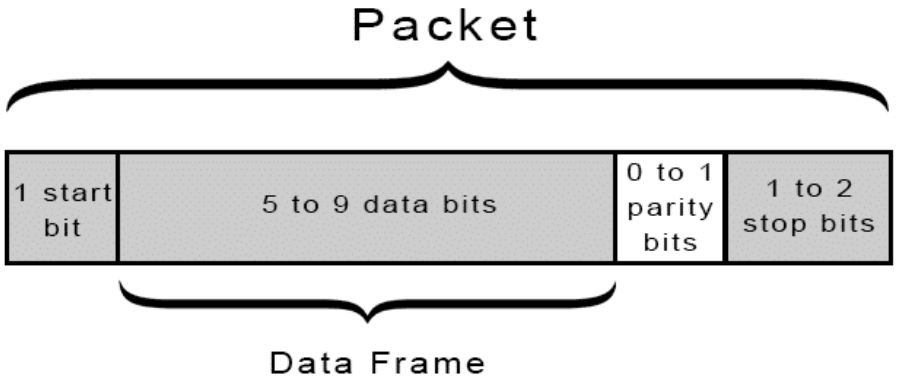


Εικόνα 3.9: Ολοκληρωμένο κύκλωμα UART. [ε18]

Τα *UART* υπάρχουν ως αυτόνομα ολοκληρωμένα κυκλώματα, αλλά πολλές φορές απαντώνται και μέσα στους ίδιους τους μικροελεγκτές. Στο φύλλο δεδομένων του κάθε μικροελεγκτή αναγράφεται η ύπαρξη και ο αριθμός των *UART*. Για παράδειγμα το *Arduino Uno* που είναι βασισμένο στο ολοκληρωμένο *ATmega328* έχει μόνο ένα *UART* ενώ το *Arduino Mega* χτισμένο πάνω σε ένα *ATmega2560* έχει τέσσερα *UART*.

Να σημειωθεί ότι στην ασύγχρονη επικοινωνία δεν υπάρχει σήμα ρολογιού για τον συγχρονισμό της εξόδου *bit* από τη μετάδοση *UART* έως τη δειγματοληψία των *bits* από το *UART* λήψης. Αντί για ένα σήμα ρολογιού το *UART* εκπομπής προσθέτει ένα *bit* έναρξης και διακοπής στο πακέτο δεδομένων που μεταφέρεται. Αυτά τα bits καθορίζουν την αρχή και το τέλος του πακέτου δεδομένων έτσι ώστε το *UART* που λαμβάνει να γνωρίζει πότε να ξεκινήσει να διαβάζει τα bits. Όταν το *UART* λήψης ανιχνεύσει το *bit* έναρξης αρχίζει να διαβάζει τα εισερχόμενα *bit* σε μια συγκεκριμένη συχνότητα γνωστή ως *baudrate*. Ο ρυθμός αυτός είναι το μέτρο της ταχύτητας μεταφοράς δεδομένων εκφραζόμενο σε *bits* ανά δευτερόλεπτο «*bps*». Και τα δύο

UART πρέπει να λειτουργούν με τον ίδιο ρυθμό *baud*. Ο ρυθμός *baud* μεταξύ των *UART* μετάδοσης και λήψης μπορεί να διαφέρει μόνο κατά περίπου 10%. Τα δεδομένα που μεταδίδονται από το *UART* είναι οργανωμένα σε πακέτα και κάθε πακέτο περιέχει 1 *bit* έναρξης, 5 έως 9 *bit* δεδομένων (ανάλογα με το *UART*), 1 προαιρετικό bit ισοτιμίας και 1 ή 2 *bit* διακοπής και η σχηματική αναπαράσταση αυτού είναι όπως στην παρακάτω **Εικόνα 3.10**.



Εικόνα 3.10: Πακέτα UART. [ε18]

Αρχικό bit

Η γραμμή μετάδοσης δεδομένων *UART* διατηρείται κανονικά σε επίπεδο υψηλής τάσης όταν δεν μεταδίδει δεδομένα. Για να ξεκινήσει η μεταφορά δεδομένων ο πομπός *UART* θέτει τη γραμμή μετάδοσης από υψηλή σε χαμηλή για έναν κύκλο ρολογιού. Όταν το *UART* λήψης ανιχνεύει τη μετάβαση από υψηλή σε χαμηλή τάση τότε αρχίζει να διαβάζει τα *bit* στο πλαίσιο δεδομένων στη συχνότητα του ρυθμού μετάδοσης.

Πλαίσιο δεδομένων

Το πλαίσιο δεδομένων περιέχει τα πραγματικά δεδομένα που μεταφέρονται. Μπορεί να έχει μήκος από 5 *bit* έως 8 *bit* εάν χρησιμοποιείται *bit* ισοτιμίας. Εάν δεν χρησιμοποιούνται τα *bit* ισοτιμίας τότε το πλαίσιο δεδομένων μπορεί να έχει μήκος έως και 9 *bit*. Στις περισσότερες περιπτώσεις, τα δεδομένα αποστέλλονται πρώτα με το λιγότερο σημαντικό *bit*. Το λιγότερο σημαντικό *bit* είναι είτε το αριστερό είτε το δεξιότερο *bit* στους δυαδικούς αριθμούς ανάλογα με την αρχιτεκτονική του υπολογιστή. Εάν το *LSB* είναι στα δεξιά, η αρχιτεκτονική ονομάζεται «*Little Endian*». Εάν το *LSB* βρίσκεται στα αριστερά, η αρχιτεκτονική ονομάζεται «*Big Endian*».

Ισοτιμία

Το *bit* ισοτιμίας είναι ένας τρόπος για το *UART* που λαμβάνει να γνωρίζει εάν άλλαξαν δεδομένα κατά τη μετάδοση. Τα *bit* μπορούν να αλλάξουν με ηλεκτρομαγνητική ακτινοβολία, ασυγχρόνιστους ρυθμούς μετάδοσης ή λόγω μεταφοράς δεδομένων σε μεγάλες αποστάσεις. Αφού το *UART* που λαμβάνει

διαβάσει το πλαίσιο δεδομένων στην συνέχεια μετρά τον αριθμό των *bit* με τιμή 1 και ελέγχει εάν το σύνολο είναι ζυγό ή μονό. Εάν το *bit* ισοτιμίας είναι 0 τότε τα 1 *bit* στο πλαίσιο δεδομένων θα πρέπει να είναι συνολικά σε ζυγό αριθμό. Εάν το *bit* ισοτιμίας είναι 1 σε αυτή την περίπτωση τα 1 *bit* στο πλαίσιο δεδομένων θα πρέπει να είναι συνολικά σε μονός αριθμός. Όταν το *bit* ισοτιμίας ταιριάζει με τα δεδομένα, το *UART* γνωρίζει ότι η μετάδοση ήταν χωρίς σφάλματα.

Τελικό bit

Το τέλος του πακέτου δεδομένων σηματοδοτείται από το *UART* που στέλνει, το οποίο οδηγεί την γραμμή μετάδοσης δεδομένων από χαμηλή τάση σε υψηλή τάση για τουλάχιστον δύο κύκλους ρολογιού.

Πλεονεκτήματα

- Χρησιμοποιούν μόνο δύο καλώδια
- Δεν απαιτείται εξωτερικό σήμα ρολογιού
- Υπάρχει έλεγχος σφαλμάτων με το *bit* ισοτιμίας
- Ευρέως χρησιμοποιούμενη μέθοδος

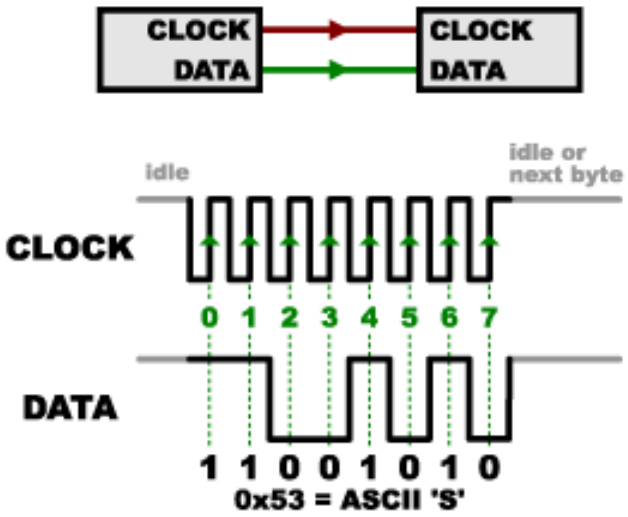
Μειονεκτήματα

- Το μέγεθος δεδομένων πλαισίου μπορεί να είναι το πολύ 9 *bit*
- Δεν υποστηρίζει συστήματα πολλαπλών συσκευών
- Μεταξύ των συσκευών επιτρέπεται το πολύ 10% απόκλιση του συγχρονισμού τους

3.4 SPI

Το *Serial Peripheral Interface* (*SPI*) [δ16] είναι ένας δίαυλος διασύνδεσης που χρησιμοποιείται συνήθως για την αποστολή δεδομένων μεταξύ μικροελεγκτών και μικρών περιφερειακών όπως καταχωρητές μετατόπισης, αισθητήρες και κάρτες *SD*. Χρησιμοποιεί ξεχωριστές γραμμές ρολογιού και δεδομένων μαζί με μια γραμμή επιλογής για να επιλεχθεί η συσκευή η οποία θα γίνει η επικοινωνία.

Πρόκειται για ένα «σύγχρονο» δίαυλο δεδομένων που σημαίνει ότι χρησιμοποιεί ξεχωριστές γραμμές για δεδομένα και ένα κοινό "ρολόι" που διατηρεί και τις δύο πλευρές σε τέλειο συγχρονισμό όπως φαίνεται στην **Εικόνα 3.11**. Το ρολόι είναι ένα ταλαντωτικό σήμα που λέει στον δέκτη ακριβώς πότε να διαβάσει τα *bit* στη γραμμή δεδομένων. Αυτό θα μπορούσε να είναι η άνοδος (χαμηλή προς υψηλή τάση) ή πτώση (υψηλή προς χαμηλή τάση) του σήματος ρολογιού. Όταν ο δέκτης εντοπίσει αυτό το άκρο τότε κοιτάζει αμέσως τη γραμμή δεδομένων για να διαβάσει το επόμενο *bit*. Επειδή το σήμα του ρολογιού αποστέλλεται μαζί με τα δεδομένα ο καθορισμός της ταχύτητας δεν είναι σημαντικός, παρόλα αυτά οι συσκευές θα έχουν μέγιστη ταχύτητα στην οποία μπορούν να λειτουργήσουν. Να σημειωθεί ότι λόγω των σημάτων υψηλής ταχύτητας το *SPI* θα πρέπει να χρησιμοποιείται μόνο για την αποστολή δεδομένων σε μικρές αποστάσεις. Εάν πρέπει να στείλουν δεδομένα πιο μακριά τότε θα πρέπει να μειωθεί σημαντικά η ταχύτητα του ρολογιού και να χρησιμοποιηθούν εξειδικευμένα τσιπ προγράμματος οδήγησης.



Εικόνα 3.11: Λειτουργία μεταφοράς δεδομένων στο SPI. [ε19]

Ένας λόγος που το *SPI* είναι τόσο δημοφιλές είναι ότι το υλικό λήψης μπορεί να είναι ένας απλός καταχωρητής. Αυτό είναι ένα πολύ απλούστερο και φθηνότερο εξάρτημα από ότι ένα πλήρες *UART* (*Universal Asynchronous Receiver / Transmitter*) που απαιτεί ασύγχρονη επικοινωνία.

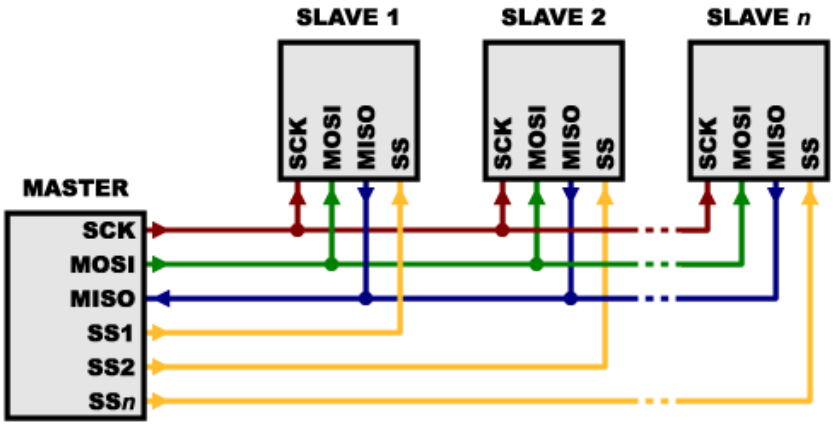
Στο *SPI* μόνο η μία πλευρά παράγει το σήμα ρολογιού και συνήθως ονομάζεται *CLK* ή *SCK* για σειριακό ρολόι. Η πλευρά που δημιουργεί το ρολόι ονομάζεται «*master*» και η άλλη πλευρά ονομάζεται «*slave*». Υπάρχει πάντα μόνο ένας κύριος αλλά μπορεί να υπάρχουν πολλοί σκλάβοι. Όταν τα δεδομένα αποστέλλονται από τον κύριο σε έναν σκλάβο αποστέλλονται σε μια γραμμή δεδομένων που ονομάζεται *MOSI (Master Out / Slave In)*. Εάν ο σκλάβος επιθυμεί να στείλει μια απάντηση πίσω στον κύριο, τότε ο κύριος θα συνεχίσει να δημιουργεί έναν προκαθορισμένο αριθμό κύκλων ρολογιού και ο σκλάβος θα βάλει τα δεδομένα σε μια τρίτη γραμμή δεδομένων που ονομάζεται *MISO (Master In / Slave Out)*.

Επειδή ο κύριος παράγει πάντα το σήμα ρολογιού, πρέπει να γνωρίζει εκ των προτέρων πότε ένας σκλάβος πρέπει να επιστρέψει δεδομένα και πόσα δεδομένα θα επιστραφούν. Αυτό είναι πολύ διαφορετικό από την ασύγχρονη σειριακή επικοινωνία όπου τυχαίες ποσότητες δεδομένων μπορούν να σταλούν σε οποιαδήποτε κατεύθυνση ανά πάσα στιγμή. Στην πράξη αυτό δεν είναι πρόβλημα καθώς το *SPI* χρησιμοποιείται γενικά για να μιλήσει με αισθητήρες που έχουν πολύ συγκεκριμένη δομή εντολών. Για παράδειγμα, εάν σταλεί η εντολή για «ανάγνωση δεδομένων» σε μια συσκευή τότε είναι γνωστό ότι η συσκευή θα στέλνει πάντα για παράδειγμα δύο *byte* σε αντάλλαγμα. Σε περιπτώσεις όπου είναι αναγκαία η επιστροφή μιας μεταβλητής ποσότητας δεδομένων, θα μπορούσε πάντα να επιστρέφεται ένα ή δύο byte που καθορίζουν το μέγιστο μήκος των δεδομένων και στη συνέχεια ο «*master*» να εξάγει τις επιθυμητές πληροφορίες στην μεριά του. Να σημειωθεί ότι το *SPI* είναι μία πλήρως αμφίδρομη επικοινωνία καθώς έχει ξεχωριστές γραμμές αποστολής και λήψης δεδομένων και επομένως είναι δυνατή η μετάδοση και η λήψη δεδομένων ταυτόχρονα.

Υπάρχει μια τελευταία γραμμή που ονομάζεται *SS (Slave Select)*. Η γραμμή αυτή λέει στον σκλάβο ότι πρέπει να ξυπνήσει και να λάβει ή να αποστείλει δεδομένα, χρησιμοποιείται επίσης όταν υπάρχουν πολλοί σκλάβοι για να επιλεχθεί ένας από τους υπόλοιπους. Η γραμμή *SS* διατηρείται συνήθως σε υψηλή τάση η οποία αποσυνδέει τον «*slave*» από το δίαυλο *SPI*. Αμέσως πριν από την αποστολή

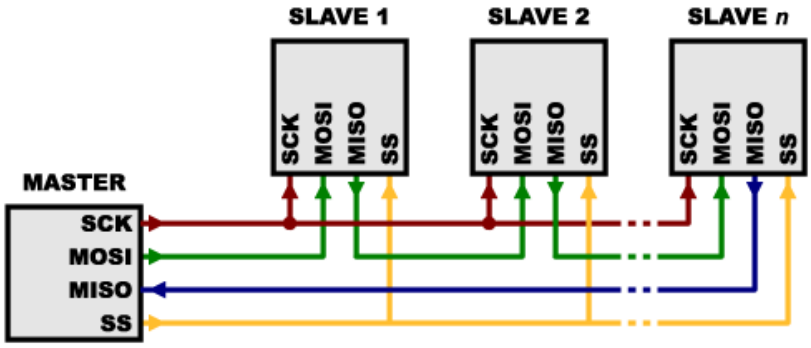
δεδομένων η γραμμή θέτεται μηδέν και ενεργοποιείται ο σκλάβος. Όταν τελειώσει ο «*master*» με τη χρήση του σκλάβου η γραμμή γίνεται ξανά ψηλή. Υπάρχουν δύο τρόποι σύνδεσης πολλών σκλάβων σε μία επικοινωνία *SPI*:

1. Γενικά ο κάθε σκλάβος θα χρειαστεί μια ξεχωριστή γραμμή *SS* όπως στην **Εικόνα 3.12**. Για να ξεκινήσει η επικοινωνία με έναν συγκεκριμένο σκλάβο θα πρέπει να τεθεί σε μηδενική τάση η κατάλληλη γραμμή *SS* του «*slave*» και να διατηρηθούν οι υπόλοιπες ψηλά θα ενεργοποιηθούν ταυτόχρονα δύο σκλάβοι ή μπορεί και οι δύο να προσπαθήσουν να μιλήσουν στην ίδια γραμμή *MISO* που όπου το αποτέλεσμα θα είναι να προκύψουν σφάλμα στα δεδομένα.



Εικόνα 3.12: Παράλληλη σύνδεση σκλάβων. [ε19]

2. Από την άλλη πλευρά είναι δυνατόν να γίνει αλυσιδωτή σύνδεση μεταξύ των σκλάβων με την *MISO* (έξοδος) του ενός να πηγαίνει στην *MOSI* (είσοδος) του επόμενου όπως στην **Εικόνα 3.13**. Σε αυτήν την περίπτωση μία κοινή γραμμή *SS* πηγαίνει σε όλους τους σκλάβους. Σημειώνεται ότι για αυτή τη διάταξη τα στοιχεία ξεχειλίζουν από έναν σκλάβο στον επόμενο. Αυτός ο τύπος διάταξης χρησιμοποιείται συνήθως σε καταστάσεις μόνο για έξοδο, όπως οδήγηση *LED* όπου δεν χρειάζεται η λήψη δεδομένων πίσω. Ωστόσο, εάν τα δεδομένα πρέπει να επιστραφούν στον «*master*» τότε αυτό είναι δυνατόν με την σύνδεση της γραμμής *MISO* του τελευταίου της αλυσίδας με τον «*master*».



Εικόνα 3.13: Αλυσιδωτή σύνδεση σκλάβων. [ε19]

Πλεονεκτήματα του SPI:

- Είναι ταχύτερο από το ασύγχρονο σειριακό.
- Το υλικό λήψης μπορεί να είναι ένας απλός καταχωρητής.
- Υποστηρίζει πολλαπλούς σκλάβους.

Μειονεκτήματα του SPI

- Απαιτεί περισσότερες γραμμές σήματος (καλώδια) από άλλες μεθόδους επικοινωνίας.
- Οι επικοινωνίες πρέπει να είναι καλά καθορισμένες εκ των προτέρων.
- Ο «*master*» πρέπει να ελέγχει όλες τις επικοινωνίες και οι σκλάβοι δεν μπορούν να μιλούν απευθείας μεταξύ τους.
- Συνήθως απαιτεί ξεχωριστές γραμμές *SS* σε κάθε «*slave*» η οποία μπορεί να είναι προβληματική εάν χρειάζονται πολλοί.

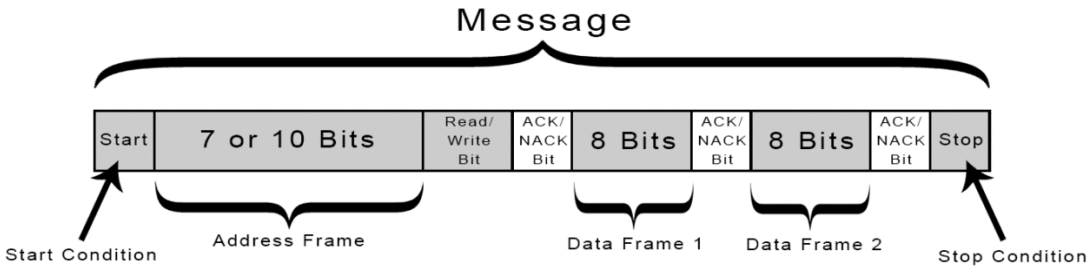
3.5 I2C

Το *I2C* [δ17] συνδυάζει τις καλύτερες δυνατότητες από τα *SPI* και *UART*. Με το *I2C* μπορούν να συνδεθούν πολλοί «*slaves*» σε έναν «*master*» όπως στο *SPI* και επίσης είναι δυνατόν να υπάρχουν πολλαπλοί «*masters*» που να ελέγχουν έναν ή περισσότερους «*slaves*» . Αυτό είναι χρήσιμο όταν χρειάζονται περισσότερα από έναν μικροελεγκτές που να καταγράφουν σε μία κάρτα μνήμης ή να εμφανίζετε κείμενο σε μία μόνο οθόνη *LCD*. Παρόμοια με την *UART* το *I2C* χρησιμοποιεί μόνο δύο καλώδια για την επικοινωνία:

- *SDA* (Σειριακά δεδομένα): Η γραμμή για την αποστολή και λήψη δεδομένων.
- *SCL* (Σειριακό ρολόι): Η γραμμή που φέρει το σήμα ρολογιού.

Το *I2C* είναι ένα πρωτόκολλο σειριακής επικοινωνίας, άρα τα δεδομένα μεταφέρονται κατά μήκος ενός μόνο καλωδίου της γραμμής *SDA*. Όπως το *SPI* έτσι και το *I2C* είναι συγχρονισμένο, οπότε η έξοδος των *bit* συγχρονίζεται με τη δειγματοληψία των *bit* με ένα σήμα κοινού ρολογιού που μοιράζεται μεταξύ του «*master»* και του «*slave»* που ελέγχεται πάντα από τον «master».

Με το συγκεκριμένο πρωτόκολλο επικοινωνίας τα δεδομένα μεταφέρονται σε μηνύματα τα οποία χωρίζονται σε πλαίσια δεδομένων. Κάθε μήνυμα έχει ένα πλαίσιο διεύθυνσης που περιέχει τη δυαδική διεύθυνση του σκλάβου και ένα ή περισσότερα πλαίσια δεδομένων που περιέχουν τα δεδομένα που μεταδίδονται. Το μήνυμα περιλαμβάνει επίσης συνθήκες έναρξης και διακοπής, *bit* ανάγνωσης / εγγραφής και *bit ACK / NACK* μεταξύ κάθε μεταδιδόμενου πλαισίου δεδομένων και παρουσιάζεται στην παρακάτω **Εικόνα 3.14**.



Εικόνα 3.14: Μήνυμα αποστολής στο πρωτόκολλο επικοινωνίας I2C. [ε20]

- Κατάσταση εκκίνησης: Η γραμμή *SDA* αλλάζει από επίπεδο υψηλής τάσης σε επίπεδο χαμηλής τάσης πριν η γραμμή *SCL* αλλάξει από υψηλή σε χαμηλή.
- Κατάσταση διακοπής: Η γραμμή *SDA* αλλάζει από επίπεδο χαμηλής τάσης σε επίπεδο υψηλής τάσης μετά τη μετάβαση της γραμμής *SCL* από χαμηλή σε υψηλή.
- Πλαίσιο διευθύνσεων: Μια ακολουθία από 7 ή 10 *bit* που είναι μοναδική για κάθε «*slave*» και προσδιορίζει την συσκευή που ο «*master*» θέλει να μιλήσει.
- *Read / Write bit*: Ένα *bit* που καθορίζει εάν ο «*master*» αποστέλλει δεδομένα στον «*slave*» (τιμή 0) ή ζητά δεδομένα από αυτόν (τιμή 1).
- *ACK / NACK bit*: Κάθε πλαίσιο σε ένα μήνυμα ακολουθείται από *bit* αναγνώρισης / μη αναγνώρισης. Εάν ένα πλαίσιο διεύθυνσης ή ένα πλαίσιο δεδομένων διαβαστεί με επιτυχία τότε ένα *bit ACK* επιστρέφεται στον αποστολέα από τη συσκευή λήψης.

Το *I2C* δεν έχει γραμμές επιλογής σκλάβων όπως το *SPI*, οπότε χρειάζεται άλλος τρόπος για να ενημερώσει τον σκλάβο ότι τα δεδομένα αποστέλλονται σε αυτόν και όχι άλλον. Αυτό είναι ξεκάθαρο με το πλαίσιο διεύθυνσης που είναι πάντα το πρώτο πλαίσιο μετά το bit έναρξης σε ένα νέο μήνυμα. Ο «*master*» στέλνει τη διεύθυνση του σκλάβου με τον οποίο θέλει να επικοινωνήσει σε κάθε σκλάβο που είναι συνδεδεμένος σε αυτόν. Στην συνέχεια κάθε σκλάβος συγκρίνει τη διεύθυνση που αποστέλλεται από τον κύριο με τη δική του διεύθυνση και εάν η διεύθυνση ταιριάζει τότε στέλνει ένα *bit ACK* χαμηλής τάσης πίσω. Εάν η διεύθυνση δεν ταιριάζει, ο σκλάβος δεν κάνει τίποτα και η γραμμή *SDA* παραμένει υψηλή.

Το πλαίσιο διευθύνσεων περιλαμβάνει ένα μόνο *bit* στο τέλος που ενημερώνει τον σκλάβο εάν ο κύριος θέλει να γράψει δεδομένα σε αυτόν ή να λάβει δεδομένα από αυτόν. Στην περίπτωση που ο κύριος θέλει να στείλει δεδομένα στον υποτελή τότε το *bit* ανάγνωσης / εγγραφής είναι 0. Στην αντίθετη περίπτωση αιτήματος δεδομένων από τον *slave* το bit είναι 1.

Αφού ο κύριος εντοπίσει το *bit ACK* από τον «*slave*», το πρώτο πλαίσιο δεδομένων είναι έτοιμο για αποστολή. Το πλαίσιο δεδομένων έχει πάντα 8 *bit* και αποστέλλεται πρώτα με το πιο σημαντικό *bit*. Κάθε πλαίσιο δεδομένων ακολουθείται αμέσως από ένα *bit ACK / NACK* για να επιβεβαιωθεί ότι το πλαίσιο έχει ληφθεί με επιτυχία. Το *bit ACK* πρέπει να ληφθεί είτε από τον κύριο είτε από τον υποτελή (ανάλογα με το ποιος στέλνει τα δεδομένα) πριν από την αποστολή του επόμενου πλαισίου δεδομένων. Μετά την αποστολή όλων των πλαισίων δεδομένων, ο κύριος μπορεί να στείλει μια κατάσταση διακοπής στον υποτελή για να σταματήσει τη μετάδοση. Η κατάσταση διακοπής είναι μια μετάβαση τάσης από χαμηλή σε υψηλή

στη γραμμή *SDA* μετά από χαμηλή σε υψηλή μετάβαση στη γραμμή *SCL*, με τη γραμμή *SCL* να παραμένει υψηλή.

Πολλοί κύριοι μπορούν να συνδεθούν με έναν μόνο σκλάβο ή πολλούς σκλάβους. Το πρόβλημα με πολλαπλούς κύριος στο ίδιο σύστημα έρχεται όταν δύο κύριοι προσπαθούν να στείλουν ή να λάβουν δεδομένα ταυτόχρονα μέσω της γραμμής *SDA*. Για την επίλυση αυτού του προβλήματος, κάθε κύριος πρέπει να εντοπίσει εάν η γραμμή SDA είναι χαμηλή ή υψηλή πριν μεταδώσει ένα μήνυμα. Εάν η γραμμή *SDA* είναι χαμηλή, αυτό σημαίνει ότι ένας άλλος κύριος έχει τον έλεγχο του διαύλου και ο κύριος θα πρέπει να περιμένει για να στείλει το μήνυμα. Εάν η γραμμή *SDA* είναι υψηλή, τότε είναι ασφαλές να μεταδοθεί το μήνυμα. Για να συνδεθούν πολλοί κύριοι σε πολλούς σκλάβους χρησιμοποιείται το διάγραμμα όπως φαίνεται στην εικόνα με 4,7*KOhm* αντιστάσεις που συνδέουν τις γραμμές *SDA* και *SCL* με *Vcc*.

Πλεονεκτήματα

- Χρησιμοποιεί μόνο δύο καλώδια.
- Υποστηρίζει πολλαπλούς «masters» και «slaves»
- Το *bit ACK / NACK* επιβεβαιώνει ότι κάθε πλαίσιο μεταφέρεται επιτυχώς
- Το υλικό είναι λιγότερο περίπλοκο σε σύγκριση με τα *UART*
- Γνωστό και ευρέως χρησιμοποιούμενο πρωτόκολλο.

Μειονεκτήματα

- Χαμηλότερος ρυθμός μεταφοράς δεδομένων από το *SPI*.
- Το μέγεθος του πλαισίου δεδομένων περιορίζεται σε 8 *bit*.
- Απαιτείται πιο περίπλοκο ηλεκτρονικό υλικό για την εφαρμογή του από ότι το *SPI*.

4.ΗΛΕΚΤΡΟΝΙΚΟ ΚΥΚΛΩΜΑ

4.1 Γενικά

Το ηλεκτρικό κύκλωμα αποτελεί μία διάταξη από ένα σύνολο ηλεκτρικών στοιχείων συνδεδεμένων μεταξύ τους στα οποία κυκλοφορεί ηλεκτρικό ρεύμα διάμεσο των αγωγών [δ18]. Βασικά ηλεκτρικά στοιχεία:

- Ηλεκτρικές πηγές
- Αντιστάσεις
- Πυκνωτές
- Πηνία

Τα ηλεκτρικά στοιχεία διακρίνονται σε δύο βασικές κατηγορίες:

- Τα ενεργά στοιχεία τα οποία παρέχουν ηλεκτρική ενέργεια στο κύκλωμα
- Τα παθητικά στοιχεία τα οποία καταναλώνουν ηλεκτρική ενέργεια

Στην περίπτωση όπου ένα στοιχείο καταναλώνει ηλεκτρική ενέργεια η οποία μετατρέπεται σε θερμότητα τότε το στοιχείο αυτό παρουσιάζει αντίσταση ηλεκτρικού ρεύματος. Το στοιχείο παρουσιάζει χωρητικότητα όταν η ηλεκτρική ενέργεια αποθηκεύεται ως ενέργεια ηλεκτρικού πεδίου. Αντίστοιχα στην αυτεπαγωγή η ηλεκτρική ενέργεια αποθηκεύεται ως ενέργεια μαγνητικού πεδίου. Να σημειωθεί ότι σε ένα πραγματικό ηλεκτρικό κύκλωμα τα στοιχεία παρουσιάζουν συνδυασμό των παραπάνω ιδιοτήτων. Στοιχεία του ηλεκτρικού κυκλώματος:

- Είσοδος ενός κυκλώματος χαρακτηρίζονται τα άκρα στα οποία επενεργεί ηλεκτρική τάση η οποία διεγείρει το κύκλωμα
- Έξοδος χαρακτηρίζονται τα δύο άκρα στα οποία μπορεί να ληφθεί τάση ή ρεύμα.
- Κλάδος, είναι το τμήμα στο οποία τα ηλεκτρικά στοιχεία διαρρέονται από της ίδιας έντασης ρεύμα.
- Κόμβος είναι το κοινό σημείο διασύνδεσης δύο ή περισσότερων ηλεκτρικών στοιχείων.
- Βρόγχος κυκλώματος είναι η αλληλουχία κλάδων που σχηματίζουν ένα κλειστό κύκλωμα.

Νόμος του Ohm

Ηλεκτρική αντίσταση είναι η ιδιότητα των υλικών να δυσκολεύουν την ροή του ηλεκτρικού ρεύματος και να μεταβάλουν την ηλεκτρική ενέργεια σε θερμότητα. Η χαρακτηριστική καμπύλη τάσης ρεύματος για την αντίσταση είναι γραμμική. Από τον νόμο του Ohm η διαφορά δυναμικού στα άκρα της αντίστασης δίνεται από την σχέση:

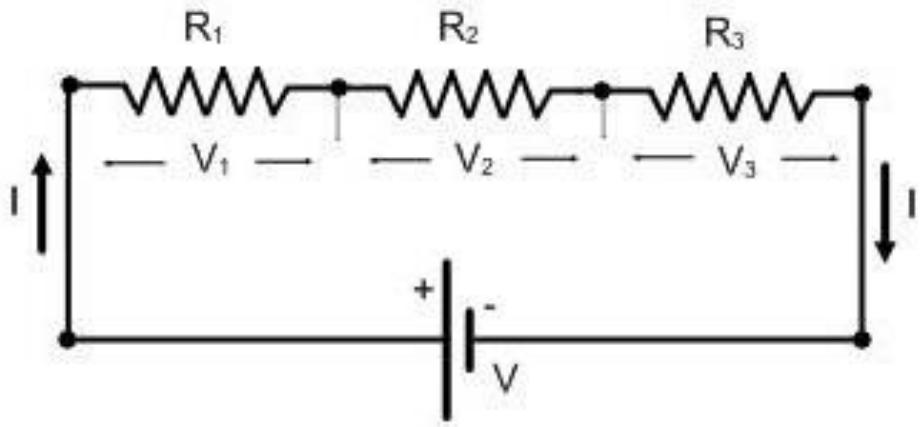
(4.1.1)

Ως αγωγιμότητα ορίζεται η ευκολία με την οποία μπορεί να περάσει το ρεύμα διαμέσου ενός υλικού και είναι το αντίστροφο της αντίστασης. Υπολογίζεται από τον τύπο:

– (4.1.2)

Σύνδεση αντιστάσεων σε σειρά

Ένας από τους πιο συνηθισμένους τρόπους σύνδεσης των αντιστάσεων είναι αυτός σε σειρά όπου όλες οι αντιστάσεις διανύονται από της ίδιας έντασης ηλεκτρικού ρεύματος όπως φαίνεται στην **Εικόνα 4.1**.



Εικόνα 4.1: σύνδεση αντιστάσεων σε σειρά. [ε21]

Στην εικόνα παρουσιάζεται το ηλεκτρικό κύκλωμα και από τον νόμο τάσεων του Kirchhoff, ότι το αλγεβρικό άθροισμα όλων των τάσεων σε κάθε βρόχο ενός κυκλώματος ισούται με μηδέν:

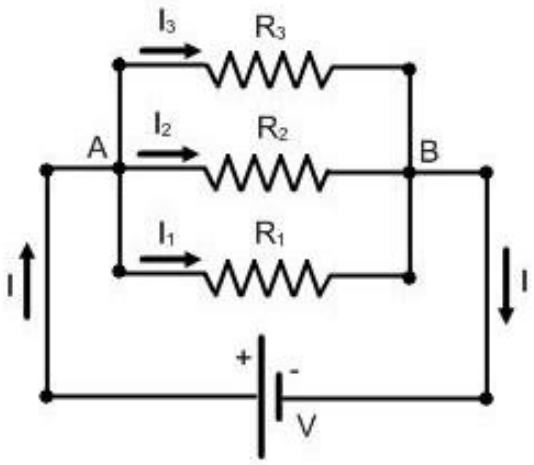
(4.1.3)

Και μέσω της σχέσης υπολογίζεται η ισοδύναμη αντίσταση κυκλώματος:

(4.1.4)

Παράλληλη σύνδεση αντιστάσεων

Περισσότερες από μία αντιστάσεις μπορούν να συνδεθούν μεταξύ τους παράλληλα όπως φαίνεται στο κύκλωμα που παρουσιάζεται στην **Εικόνα 4.2**. Σε αυτή την περίπτωση η διαφορά δυναμικού στα άκρα κάθε αντίστασης είναι ίδια.



Εικόνα 4.2: Παράλληλη σύνδεση αντιστάσεων. [ε21]

Από τον νόμο των ρευμάτων του Kirchhoff ισχύει ότι το αλγεβρικό άθροισμα όλων των ρευμάτων σε κάθε κόμβο του κυκλώματος ισούται με μηδέν, περιγράφεται από την σχέση:

(4.1.5)

Από την σχέση υπολογίζεται η ισοδύναμη αντίσταση σε παράλληλη σύνδεση από την σχέση:

— — (4.1.6)

Διαιρέτης τάσης

Με τον διαιρέτη τάσης επιτυγχάνεται η δημιουργία ενδιάμεσων τιμών τάσης από αυτής της πηγής Vs που μπορεί να απαιτούνται σε ένα κύκλωμα. Αυτό επιτυγχάνεται μέσω ενός συνόλου αντιστάσεων συνδεδεμένες σε σειρά και όπως είναι γνωστό κάθε αντίσταση έχει μία μικρότερη πτώση τάσης στα άκρα της σε σχέση με την πηγή. Η συγκεκριμένη τιμή τάσης που εφαρμόζεται στα άκρα μίας αντίστασης συνδεδεμένη σε σειρά με άλλες αντιστάσεις υπολογίζεται από την σχέση:

——— (4.1.7)

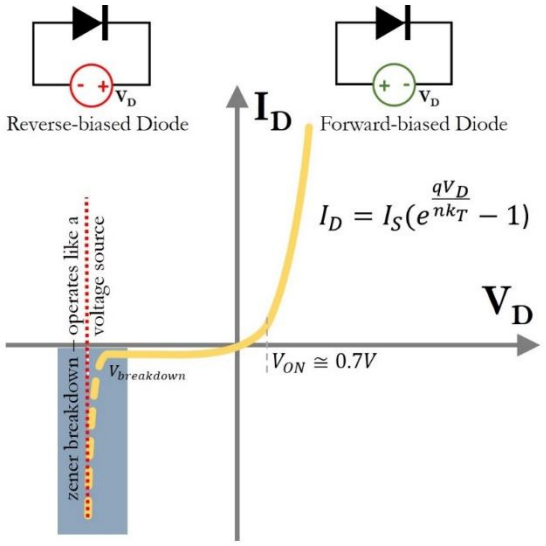
Με την επιλογή κατάλληλων τιμών αντιστάσεων είναι δυνατό οποιοδήποτε υποπολλαπλάσιο της τάσης της πηγής. Να σημειωθεί ότι η σχέση που περιγράφει το διαιρέτη τάσης ισχύει μόνο στην περίπτωση όπου δεν υπάρχει τίποτε άλλο συνδεδεμένο στους ενδιάμεσους κόμβους του κυκλώματος.

Διαιρέτης ρεύματος

Αντίστοιχα μία ή περισσότερες αντιστάσεις συνδεδεμένες παράλληλα έχουν ως αποτέλεσμα κάθε αντίσταση μεμονωμένα να διαρρέεται από υποπολλαπλάσιο ρεύμα του συνολικού ρεύματος τροφοδοσίας της πηγής Is. Κάθε ένταση ρεύματος υπολογίζεται από την σχέση:

(4.1.8)

Έτσι οποιαδήποτε υποπολλαπλάσια ένταση ρεύματος είναι εφικτή με την κατάλληλη επιλογή τιμών αντίστασης σε παράλληλη σύνδεση. Ηλεκτρονικό κύκλωμα χαρακτηρίζεται το κύκλωμα στο οποίο εκτός από αντιστάτες, πηνία και πυκνωτές διαθέτει και ηλεκτρονικά στοιχεία ημιαγωγών όπως διόδους, τρανζίστορ, τελεστικούς ενισχυτές. Το βασικό στοιχείο όλων τον ηλεκτρονικών στοιχείων είναι το τρανζίστορ το οποίο αποτελείται από δύο διόδους. Η δίοδος είναι ένα στοιχείο το οποίο επιτρέπει την διέλευση του ηλεκτρικού ρεύματος μονοσήμαντα και μπλοκάρει την κίνηση στην αντίθετη κατεύθυνση. Οι περισσότερες σύγχρονες δίοδοι βασίζονται στον ημιαγωγό p-n όπου στο σημείο επαφής των ημιαγωγών δημιουργείται μία στενή ζώνη ονομαζόμενη περιοχή φορτίου χώρου όπου δημιουργείται το λεγόμενο φράγμα δυναμικού. Η δίοδος θεωρείται ορθά πολωμένη όταν ο θετικός πόλος της πηγής τροφοδοσίας συνδέεται με την περιοχή p και αντίστοιχα ο αρνητικός με την περιοχή n. Η διέλευση ρεύματος ξεκινά όταν η τάσης της πηγής υπερβαίνει την τάση του φράγματος δυναμικού ή τάση κατωφλιού όπως αλλιώς ονομάζεται. Η χαρακτηριστική ρεύματος-τάσης της διόδου παρουσιάζεται στην **Εικόνα 4.3** και περιγράφεται από την σχέση.



Εικόνα 4.3: Χαρακτηριστική ρεύματος-τάσης διόδου. [ε22]

(4.1.9)

όπου

, φορτίο ηλεκτρονίου

n = σταθερά

k = σταθερά Boltzman

T = απόλυτη θερμοκρασία

Η αντίσταση σώματος r_B της διόδου δίνεται από την σχέση και αποτελεί την πραγματική ωμική αντίσταση της διόδου.

—— (4.1.10)

Για τον σχεδιασμό του ηλεκτρονικού κυκλώματος του σαρωτή θα πρέπει να ληφθεί υπόψη ποιες είναι οι λειτουργίες που επιτελούνται καθώς και με ποια ηλεκτρονικά στοιχεία είναι εφικτό αυτό. Το ηλεκτρονικό σύστημα καλείται:

- Να περιστρέφει την τράπεζα και να κινεί τον αισθητήρα *TOF* όταν δέχεται την κατάλληλη είσοδο από τον χρήστη με το πάτημα ενός πλήκτρου.
- Να διαθέτει διακόπτες ενεργοποίησης-απενέργοποίησης τροφοδοσίας του συνολικού κυκλώματος

Για την περιστροφή της τράπεζας και την κίνηση του αισθητήρα απόστασης στον άξονα *Z* θα χρησιμοποιηθούν βηματικοί κινητήρες καθώς είναι απαραίτητο να ελέγχεται η ταχύτητα και η θέση περιστροφής. Η επιλογή του βηματικού κινητήρα αποτελεί την απλούστερη και φθηνότερη λύση για εφαρμογές χαμηλής ροπής. Η οδήγηση του κινητήρα θα γίνεται μέσω ηλεκτρονικού κυκλώματος οδήγησης που είναι διαθέσιμο στο εμπόριο. Το κύκλωμα οδήγησης του βηματικού δέχεται ως είσοδο σήματα ελέγχου τα οποία παράγονται από έναν προγραμματιζόμενο ελεγκτή ο οποίος θα εκτελεί εντολές σύμφωνα με τα σήματα εισόδου από τον χρήστη. Όλα τα ηλεκτρονικά στοιχεία θα στεγάζονται στην ηλεκτρονική πλακέτα όπου και θα τροφοδοτούνται με ηλεκτρικό ρεύμα στην επιθυμητή τάση και θα συνδέονται μεταξύ τους κατάλληλα. Ο κεντρικός ελεγκτής θα έχει προγραμματιστεί έτσι ώστε να διαχειρίζεται όλες τις εισόδους και εξόδους.

4.2 Μικροεπεξεργαστές και μικροελεγκτές

Για να γίνει η σωστή επιλογή του προγραμματιζόμενου ελεγκτή θα πρέπει να γίνει κατανοητή η διαφορά μεταξύ των μικροελεγκτών και των μικροεπεξεργαστών.

Ο μικροεπεξεργαστής πρόκειται για ένα ολοκληρωμένο κύκλωμα το οποίο περιέχει μόνο την *CPU* και επομένως διαθέτει αποκλειστικά δυνατότητες επεξεργαστικής ισχύς [δ19]. Για την αποθήκευση των προγραμμάτων και δεδομένων χρησιμοποιούνται εξωτερικές *RAM* και *ROM*. Να σημειωθεί ότι μπορεί να περιέχει περισσότερους από έναν πυρήνα επιτρέποντας την λειτουργία «*multitasking*» και επιπλέον η ταχύτητα του ρολογιού τους είναι αρκετά μεγάλη. Λόγω των χαρακτηριστικών του ο μικροεπεξεργαστής βρίσκει εφαρμογή κυρίως σε συσκευές όπως οι ηλεκτρονικοί υπολογιστές ή τα κινητά και χρησιμοποιείται για ανάπτυξη ψηφιακών λογισμικών, παιχνιδιών και εφαρμογών ήχου και εικόνας.

Από την άλλη μεριά ο μικροελεγκτής είναι ένα ολοκληρωμένο κύκλωμα το οποίο διαθέτει *CPU* αλλά και καθορισμένη μνήμη *RAM* και *ROM* καθώς και όλα τα απαραίτητα περιφερειακά είναι ενσωματωμένα σε ένα μόνο τσιπ. Στην πραγματικότητα πρόκειται για έναν ολοκληρωμένο σύστημα μικροϋπολογιστή με περιορισμένους διαθέσιμους πόρους και χαμηλότερη συχνότητα ρολογιού από αυτή του μικροεπεξεργαστή. Δεν διαθέτουν δυνατότητα πραγματικού «*multitasking*» και είναι σχεδιασμένοι κυρίως για εφαρμογές πραγματικού χρόνου και σήματα εισόδου εξόδου. Βρίσκει εφαρμογή σε συσκευές όπως πλυντήρια, ψυγεία, τηλεχειριστήρια κ.α.

Πίνακας 4-1: Σύγκριση μικροεπεξεργαστών με τους μικροελεγκτές

Μικροεπεξεργαστής	Μικροελεγκτής
Υπολογιστικά συστήματα	Ενσωματωμένα συστήματα
Επεξεργαστής μόνο	Επεξεργαστής και περιφερειακά
Ακριβός	Φθηνός
Μεγάλη κατανάλωση ενέργειας	Μικρή κατανάλωση ενέργειας
Δεν συνιστάται για εφαρμογές πραγματικού χρόνου	Κατάλληλος για εφαρμογές πραγματικού χρόνου
Εκτέλεση παράλληλων διεργασιών	Εκτέλεση μίας διεργασίας

4.3 Επιλογή ελεγκτή

Το ηλεκτρονικό κύκλωμα που θα υλοποιηθεί στα πλαίσια της πτυχιακής είναι υπεύθυνο για τον έλεγχο ενός βηματικού κινητήρα που περιστρέφει την τράπεζα καθώς και των σημάτων ελέγχου από και τον χειριστή. Για την εφαρμογή αυτή θα χρησιμοποιηθεί ένα ολοκληρωμένος ελεγκτής του εμπορίου που στον πυρήνα του διαθέτει ολοκληρωμένο κύκλωμα μικροελεγκτή, καθώς δεν απαιτείται επεξεργαστική ισχύ αλλά κυρίως επεξεργασία και παραγωγή σημάτων σε πραγματικό χρόνο. Δημοφιλείς αναπτυξιακές πλακέτες παρουσιάζονται ενδεικτικά στους παρακάτω πίνακες:

Πίνακας 4-2: Δημοφιλείς αναπτυξιακές πλακέτες

Atmel AVR			
Πλακέτα	**Τσιπ**	**Πλακέτα**	**Τσιπ**
Arduino Uno	ATmega328	Arduino Mega 2560	ATmega32u4
Arduino Leonardo	ATmega32u4	Arduino Due	AT91SAM3X8E
Arduino Micro	ATmega32u4	Arduino Nano	ATmega328
ARM			
Πλακέτα	**Τσιπ**	**Πλακέτα**	**Τσιπ**
Raspberry Pi 4	BCM2711	Raspberry Pi Zero	BCM2835
Teensy 3.6	180 MHz ARM Cortex-M4	Teensy LC	48MHz ARM Cortex-M0
Beaglebone Black	AM3358 1GHz ARM® Cortex-A8	STM32 Nucleo	STM32F401RE

Η καθοδήγηση του βηματικού πραγματοποιείται μέσω των σημάτων εισόδου *STEP* και *DIR* και παράγονται από έναν προγραμματιζόμενο ηλεκτρονικό ελεγκτή. Από τις αναπτυξιακές πλακέτες που αναλύθηκαν στο κεφάλαιο επιλέγεται το *Arduino Nano (clone)* με γνώμονα:

- την απαιτούμενη υπολογιστική ισχύ
- το κόστος
- τις διαθέσιμες εισόδους εξόδους

Πρόκειται για την πιο μικρή πλακέτα της τεχνολογίας *Arduino* ως ισοδύναμο του *Arduino Uno*. Βασίζεται στον μικροελεγκτή *ATmega328* της *Atmel*. Μπορεί να προγραμματιστεί και να λειτουργήσει μέσω της θύρας *Mini-B USB*. Να σημειωθεί πως η βασική διαφορά μεταξύ της γνήσιας και της αντίγραφης πλακέτας είναι το *chip* που χρησιμοποιείται ως *USB to Serial converter* και συγκεκριμένα στην γνήσια χρησιμοποιείται το *FTDI FT232RL* ενώ στην αντίγραφη το *CH340*G. Παρουσιάζονται τα τεχνικά χαρακτηριστικά στον παρακάτω πίνακα:

Πίνακας 4-3: Χαρακτηριστικά *Arduino Nano*

ARDUINO NANO	
Μικροελεγκτής	ATmega328
Αρχιτεκτονική	AVR
Τάση λειτουργίας	5 V.
Μνήμη Flash	32 KB εκ των οποίων 2 KB χρησιμοποιούνται από το bootloader
SRAM	2 KB
Ταχύτητα ρολογιού	16 MHz
Αναλογικές καρφίτσες εισόδου / εξόδου	8
EEPROM	1 KB
DC ρεύμα ανά καρφίτσες εισόδου / εξόδου	40 mA (καρφίτσες εισόδου / εξόδου)
Τάση εισόδου	7-12 V.
Ψηφιακές καρφίτσες εισόδου / εξόδου	22
Έξοδος PWM	6
Κατανάλωση ενέργειας	19 mA
Μέγεθος PCB	18 x 45 mm
Βάρος	7 γρ.

4.4 Οδηγός κινητήρα

Όπως αναλύθηκε σε προηγούμενο κεφάλαιο ο βηματικός κινητήρας για να μπορεί να ελεγχθεί χρειάζεται ένα κύκλωμα οδήγησης το οποίο θα στείλει τις κατάλληλες τάσεις με την σωστή αλληλουχία στις φάσεις του κινητήρα δεχόμενο μόνο 2 σήματα ελέγχου από τον ηλεκτρονικό ελεγκτή. Οι πιο συνηθισμένοι οδηγοί του εμπορίου που χρησιμοποιούνται για τον έλεγχο βηματικών κινητήρων χαμηλής ισχύος είναι:

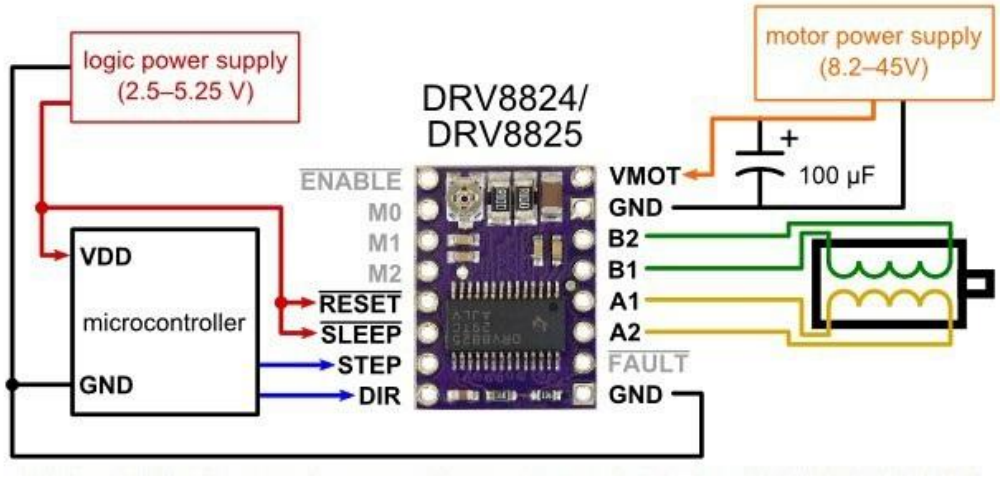
- A4988
- DRV8825
- MP6500
- TMC2100

Η ελεγχόμενη περιστροφή της τράπεζα πραγματοποιείται μέσω βηματικού κινητήρα. Σε προηγούμενο κεφάλαιο επιλέχθηκε ο βηματικός κινητήρας *NEMA17 6KG JK42HS40-1304F* όπου η κάθε φάση του έχει ονομαστική λειτουργία ρεύματος 1,2A. Με βάση αυτή την πληροφορία και το κόστος αφοράς επιλέγεται ο *DRV8825* οδηγός.

Χαρακτηριστικά DRV8825

- *PWM Microstepping*
  - Ενσωματωμένο *Microstepping Indexer*
  - Μέχρι 1/32 *Microstepping*
- Πολλαπλά *Decay Modes*
  - Ανάμεικτα *Decay*
  - Αργά *Decay*
  - Γρήγορα *Decay*
- Τροφοδοσία από 8.2V to 45V
- Μέγιστο ρεύμα 2.5A στα 24V και θερμοκρασία T_A = 25°C
- Λειτουργία *STEP/DIR*
- Λειτουργία ύπνου χαμηλού ρεύματος
- Ενσωματωμένη έξοδο αναφοράς 3.3V
- Μικρό αποτύπωμα
- Λειτουργίες προστασίας
  - Προστασία από υπερένταση (*OCP*)
  - Θερμική προστασία (*TSD*)
  - Κλείδωμα χαμηλής τάσης (*UVLO*)
  - Σήμα ένδειξης βλάβης (*nFAULT*)

Στην **Εικόνα 4.4** φαίνεται η συνδεσμολογία λειτουργίας του οδηγού που θα πρέπει να ενσωματωθεί στο καθολικό ηλεκτρονικό κύκλωμα.



Εικόνα 4.4: Συνδεσμολογία οδηγού βηματικού κινητήρα. [ε23]

4.5 Αισθητήρας TOF

Επιλέγεται ο αισθητήρας *Adafruit VL6180X* ο οποίος είναι ένας αισθητήρας απόστασης κατηγορίας *Time of Flight*. Η διάταξη περιέχει μια πολύ μικρή πηγή λέιζερ και έναν αντίστοιχο αισθητήρα που μπορεί να ανιχνεύσει τον «χρόνο πτήσης» ή πόσο χρόνο χρειάστηκε η ακτίνα λέιζερ για να επιστρέψει πίσω από την στιγμή εκπομπής. Δεδομένου ότι χρησιμοποιεί μια πολύ στενή πηγή φωτός, είναι καλός για τον προσδιορισμό της απόστασης μόνο της επιφάνειας ακριβώς μπροστά της. Σε αντίθεση με τους αισθητήρες υπερήχων όπου τα υπερηχητικά κύματα αναπηδούν, ο «κώνος» της αίσθησης είναι στενός και για την ακρίβεια 25 μοίρες. Αντίστοιχα σε σχέση με τους αισθητήρες απόστασης υπερύθρων *IR* που προσπαθούν να

μετρήσουν την αναπήδηση του φωτός, το *VL6180X* είναι πολύ πιο ακριβές και δεν έχει προβλήματα γραμμικότητας ή «διπλής απεικόνισης» όπου δεν είναι ξεκάθαρο εάν ένα αντικείμενο είναι πολύ μακριά ή πολύ κοντά. Μπορεί να χειριστεί αποστάσεις περίπου από 5mm έως 100mm αλλά με καλές συνθήκες περιβάλλοντος επιτυγχάνεται και εύρος 150-200mm. Περιλαμβάνει επίσης έναν αισθητήρα *lux* για την μέτρηση της έντασης της φωτεινότητας του περιβάλλοντος. Δεδομένου ότι το ολοκληρωμένο κύκλωμα χρειάζεται μικρή ισχύ και τάση 2.8V έχει τοποθετηθεί στο συνολικό κύκλωμα ένας ρυθμιστής τάσης έτσι ώστε μπορεί να χρησιμοποιηθεί οποιαδήποτε τάση τροφοδοσίας 3-5V που διαθέτουν οι περισσότεροι μικροελεγκτές. Η επικοινωνία με τον αισθητήρα γίνεται μέσω του πρωτοκόλλου *I2C* και με μερικές απλές εντολές καθώς το μεγαλύτερο μέρος της εργασίας γίνεται μέσα στον ίδιο τον αισθητήρα. Επιλέχθηκε ο συγκεκριμένος και φαίνεται στην παρακάτω **Εικόνα 4.5**.

Εικόνα 4.5: Αισθητήρας απόστασης Adafruit VL6180X. [ε24]

4.6 Σχεδιασμός πλακέτας

Ηλεκτρονικό σχηματικό διάγραμμα είναι το διάγραμμα στο οποίο παρουσιάζονται τα ηλεκτρονικά στοιχεία του κυκλώματος με γραφικά σύμβολα και οι φυσικές ηλεκτρονικές συνδέσεις εισόδων εξόδων, σημάτων και τροφοδοσίας με απλό και ευδιάκριτο τρόπο. Πρόκειται για τον πρόδρομο της διαδικασίας κατασκευής ενός ηλεκτρονικού κυκλώματος κατά το στάδιο στο οποίο εκτελούνται διαδικασίες υπολογισμών ρευμάτων, τάσεων, αντοχών, διεξαγωγή προσομοιώσεων, διόρθωση τυχόν λογικών σφαλμάτων και επανασχεδιασμός αν χρειαστεί. Εκτός από την κατασκευή της φυσικής πλακέτας που επακολουθεί της ολοκλήρωσης της σχεδίασης το σχηματικό είναι χρήσιμο και για την μελέτη και κατανόηση ήδη υπαρχόντων κυκλωμάτων σε περιπτώσεις εύρεσης αστοχιών και επισκευής. Να σημειωθεί ότι εκτός από τα ηλεκτρονικά τα σχηματικά βρίσκουν εφαρμογή στην μηχανολογία, την χημεία ακόμα και σε επιχειρηματικά μοντέλα.

Το ηλεκτρονικό κύκλωμα της παρούσας πτυχιακής διαθέτει τα παρακάτω βασικά ηλεκτρονικά στοιχεία που φαίνονται στην **Εικόνα 4.6**:

- Κύκλωμα οδήγησης κινητήρων *DRV8825.*
- Προγραμματιζόμενος μικροελεγκτής *Arduino Nano.*
- *SD card Breakout.*
- Αισθητήρας απόστασης *Adafruit TOF VL6180X.*
- Διακόπτης τροφοδοσίας.
- *Power Jack*

- Κουμπί στιγμιαίας σύνδεσης
- Τροφοδοτικό

Εικόνα 4.6: Βασικά ηλεκτρονικά στοιχεία κυκλώματος.

Από τα τεχνικά χαρακτηριστικά του κυκλώματος οδήγησης του βηματικού κινητήρα προτείνεται τάση τροφοδοσίας στο εύρος 8,2V έως 45V. Ο βηματικός κινητήρας *NEMA17 JK42HS40-1304F* έχει ονομαστικό ρεύμα I_M ανά φάση 1,2A. Παρόλα αυτά το ηλεκτρονικό κύκλωμα οδήγησης *DRV8825* διαθέτει έναν ροοστάτη μέσω του οποίου είναι δυνατόν να ρυθμιστεί το ρεύμα που διέρχεται από κάθε φάση και μιας η εφαρμογή δεν απαιτεί υψηλές ροπές το ρεύμα θα μειωθεί στο σημείο που η συσκευή θα είναι λειτουργική και με μικρή κατανάλωση ισχύος, I_M = 0,35A. Αντίστοιχα στα τεχνικά χαρακτηριστικά του *Arduino Nano* προτείνεται τάση τροφοδοσίας από 7V έως 12V και η μέγιστη τιμή ρεύματος τροφοδοσίας I_A είναι τα 0,5A. Ο αισθητήρας *TOF* τροφοδοτείται με ονομαστική τάση 5V από την έξοδο που διαθέτει ο μικροελεγκτής επάνω του και έχει ονομαστικό ρεύμα λειτουργίας I_T 0,04A. Επομένως η κατάλληλη τάση τροφοδοσίας του συνολικού κυκλώματος είναι τα 12V και το συνολικό απαιτούμενο φορτίο από την τροφοδοσία υπολογίζεται από τον Νόμο των ρευμάτων του Kirchhoff:

Επομένως θα χρησιμοποιηθεί τροφοδοτικό τάσης τροφοδοσίας 12V και ονομαστικού ρεύματος 2A το οποίο θα συνδέεται στο κύκλωμα μέσω *power jack* όπου στην συνέχεια θα παρεμβάλλεται ένας διακόπτης που θα ανοίγει και θα κλείνει το κύκλωμα τροφοδοσίας.

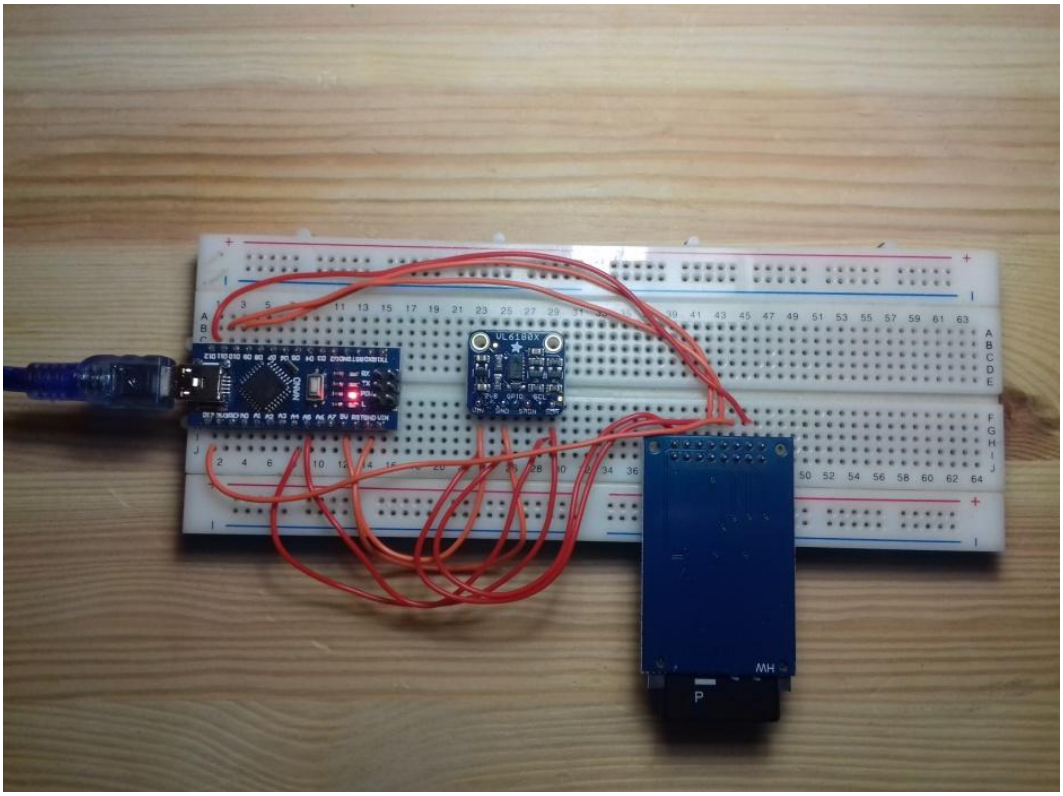
Τα σήματα εισόδου εξόδου μεταξύ των στοιχείων δεν πρέπει να υπερβαίνουν τα 5V σύμφωνα με τις τεχνικές οδηγίες, καθώς αυτό θα οδηγήσει στην καταστροφή

τους. Η διαδικασία σάρωσης με αποθήκευση στην κάρτα μνήμης θα πυροδοτείται από ένα κουμπί στιγμιαίας λειτουργίας που θα στέλνει ένα σήμα στην είσοδο του ελεγκτή και τα τροφοδοτείται με 5V από την έξοδο του ρυθμιστή τάσης του *Arduino Nano*.

Αφού έχουν υπολογιστεί τα κατάλληλα ηλεκτρονικά στοιχεία και έχουν θεσπιστεί οι μεταξύ τους συνδέσεις όπως απαιτείται, τελικό βήμα είναι να σχεδιαστεί το διάγραμμα κυκλώματος της συνολικής ηλεκτρονικής πλακέτας. Ο σχεδιασμός του ηλεκτρονικού κυκλώματος στα πλαίσια αυτής της πτυχιακής εργασίας πραγματοποιήθηκε στην δωρεάν διαδικτυακή σουίτα *EasyEDA* [δ20] και παρουσιάζεται στο ΠΑΡΑΡΤΗΜΑ Β.

4.7 Υλοποίηση πλακέτας

Πριν την τελική υλοποίηση της ηλεκτρονικής πλακέτας ένα πολύ σημαντικό στάδιο της διαδικασίας είναι η προσωρινή συναρμολόγηση για τον έλεγχο λειτουργίας του κυκλώματος επάνω σε πλακέτα δοκιμών όπως φαίνεται στην **Εικόνα 4.7**. Οι πλακέτες δοκιμών είναι κατασκευασμένες από σκληρό άκαμπτο υλικό σε ορθογώνιο σχήμα και διαθέτουν οπές όπου οι ακροδέκτες καλωδίων μπορούν να γεφυρωθούν στις επιθυμητές συνδέσεις με τα ηλεκτρονικά στοιχεία χωρίς να απαιτείται μόνιμη κόλληση. Οι οπές είναι ομαδοποιημένες συνήθως σε πεντάδες και είναι ηλεκτρικά συνδεδεμένες μεταξύ τους. Σε κάθε ομάδα υπάρχει ένα μεταλλικό έλασμα το οποίο συγκρατεί και κάνει επαφή με τους ακροδέκτες που τοποθετούνται. Κατά μήκος των δύο κάθετων σειρών της πλακέτας υπάρχουν δύο σειρές σε κάθε πλευρά και χρησιμοποιούνται ως γραμμές τροφοδοσίας.



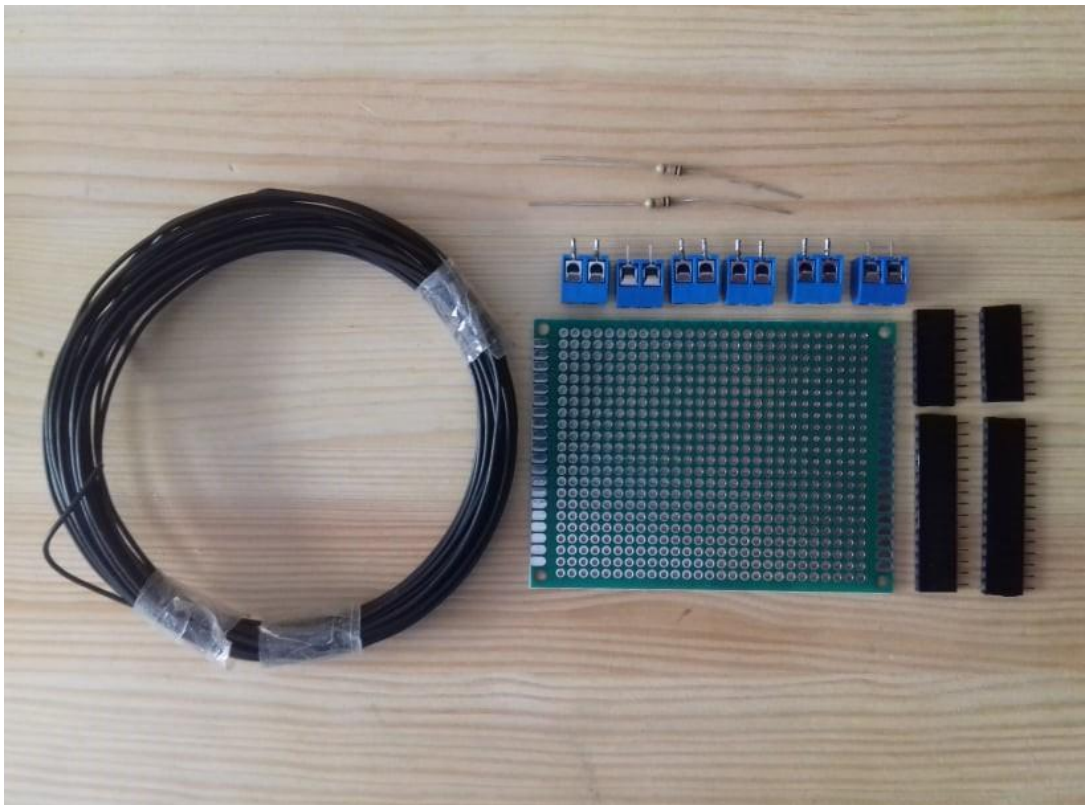
Εικόνα 4.7: Προσωρινός έλεγχος κυκλώματος σε πλακέτα δοκιμών.

Για την κατασκευή του τελικού ηλεκτρονικού κυκλώματος θα χρησιμοποιηθούν τα εργαλεία όπως φαίνονται στην **Εικόνα 4.8** και αυτά είναι:

Εικόνα 4.8: Εργαλεία υλοποίησης πλακέτας.

- Ηλεκτρικό θερμαινόμενο κολλητήρι 40W.
- Υλικό συγκόλλησης καλάι.
- Σφουγγάρι για τον καθαρισμό μύτης συγκόλλησης.
- Τρόμπα αναρρόφησης για τυχόν διορθώσεις.
- Ψηφιακό πολύμετρο.

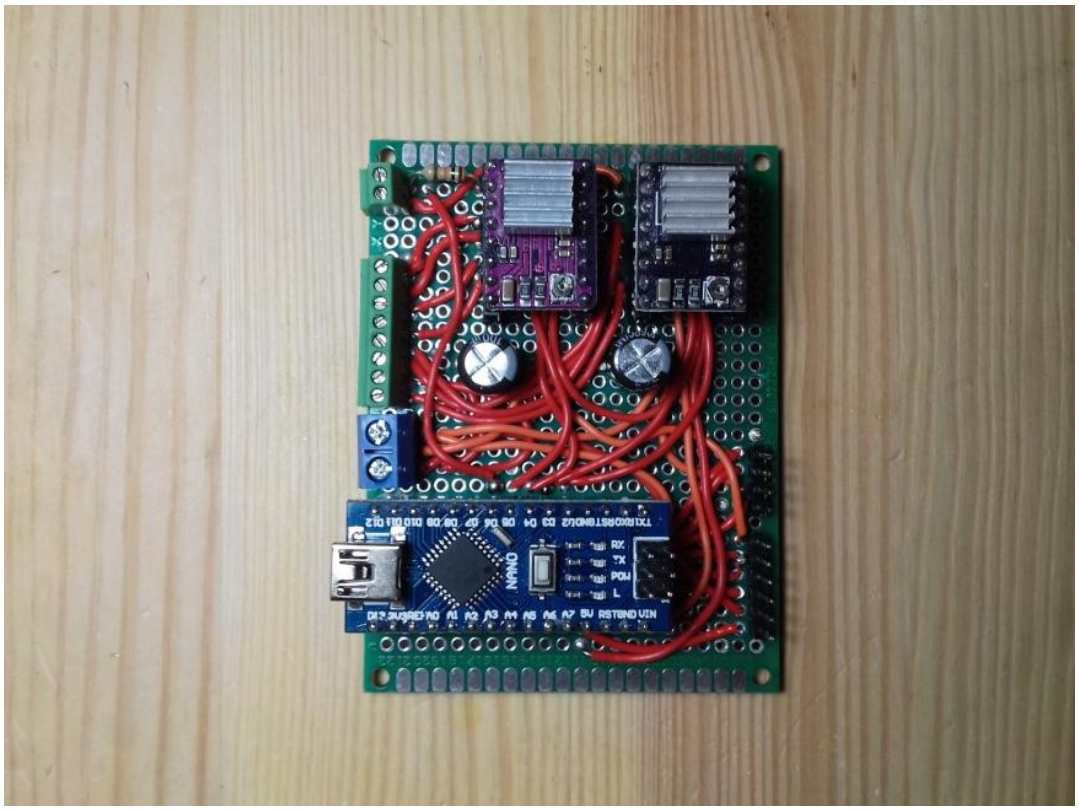
Το κύκλωμα θα κατασκευαστεί επάνω σε διατρητή πλακέτα διπλής όψης που επιτρέπει την συγκόλληση και από τις δύο επιφάνειες. Τα βασικά στοιχεία σύνδεσης του ηλεκτρονικού κυκλώματος φαίνονται στην **Εικόνα 4.9** και είναι:



Εικόνα 4.9: Βασικά στοιχεία σύνδεσης κυκλώματος.

- Διάτρητη πλακέτα κατασκευών 60x80mm.
- Τερματικά καλωδίων 2P 2,54mm.
- Υποδοχές 2x15 2,54mmγια το *Arduino Nano*.
- Υποδοχές 2x8 2,54mm για κάθε *DRV8825*.
- Ηλεκτρολυτικός πυκνωτής 50V 100uF.
- Ωμική αντίσταση 10KOhm 0,2W.
- Μονόκλωνο καλώδιο συρμάτωσης.

Αρχικά γίνονται οι κολλήσεις και προσαρμόζονται τα βασικά στοιχεία επάνω στην διάτρητη πλακέτα. Να σημειωθεί ότι είναι πολύ σημαντικό πριν συνδεθούν τα ηλεκτρονικά στοιχεία επάνω στην πλακέτα και τροφοδοτηθούν με ρεύμα είναι πολύ σημαντικό να γίνει έλεγχος πρώτα με το ψηφιακό πολύμετρο και να εντοπιστούν τυχόν βραχυκυκλώσεις ή λάθος συνδέσεις. Με αυτόν τον τρόπο θα αποφευχθεί οποιαδήποτε βλάβη στα ηλεκτρονικά στοιχεία. Σε περίπτωση ανάγκης διόρθωσης η κόλληση ξαναζεσταίνεται με το κολλητήρι και το καλάι απομακρύνεται με αναρρόφηση μέσω της τρόμπας που ενδείκνυται γι αυτό τον σκοπό. Έπειτα συνδέονται τα ηλεκτρονικά στοιχεία και η τελική ηλεκτρονική πλακέτα παρουσιάζεται στην **Εικόνα 4.10**.



Εικόνα 4.10: Τελική ηλεκτρονική πλακέτα σαρωτή.

5.ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΚΑΤΑΣΚΕΥΗ ΣΑΡΩΤΗ

5.1 Ψηφιακή σχεδίαση

Ο ηλεκτρονικός σχεδιασμός της διάταξης του σαρωτή θα πραγματοποιηθεί με την βοήθεια του λογισμικού σχεδίασης SolidWorks. Η δημιουργία ενός μοντέλου στο *SolidWorks* ξεκινά συνήθως με ένα *2D* σκίτσο. Το σχέδιο αποτελείται από σημεία, γραμμές, τόξα, κωνικά και *splines*. Οι διαστάσεις προστίθενται στο σκίτσο για να καθορίσουν το μέγεθος και τη θέση της γεωμετρίας. Χρησιμοποιούνται επίσης σχέσεις για τον καθορισμό χαρακτηριστικών όπως η εφαπτομένη, ο παραλληλισμός, η κάθετη και η ομόκεντρος. Η παραμετρική φύση του *SolidWorks* σημαίνει ότι οι διαστάσεις και οι σχέσεις οδηγούν στη γεωμετρία και όχι το αντίστροφο. Οι διαστάσεις στο σχέδιο μπορούν να ελεγχθούν ανεξάρτητα ή από σχέσεις με άλλες παραμέτρους εντός ή εκτός του σχεδίου. Σε μία συναρμογή οι αντίστοιχες σχέσεις προς το δισδιάστατο σκίτσο είναι τα «*mates*». Ακριβώς όπως οι σχέσεις σκίτσων ορίζουν συνθήκες όπως η εφαπτομένη, ο παραλληλισμός και ομόκεντρος σε σχέση με τη γεωμετρία του σκίτσου, στις συναρμογές πολλών τρισδιάστατων επιμέρους σχεδίων ορίζονται ισοδύναμες σχέσεις «*mates*», επιτρέποντας την εύκολη κατασκευή συγκροτημάτων. Το *SolidWorks* περιλαμβάνει επίσης πρόσθετα προηγμένα χαρακτηριστικά συναρμολόγησης εξαρτημάτων όπως οι σχέσεις μετάδοσης κίνησης. Τέλος, τα κατασκευαστικά σχέδια μπορούν να δημιουργηθούν είτε από εξαρτήματα είτε από συγκροτήματα. Οι προβολές δημιουργούνται αυτόματα από το συμπαγές μοντέλο και οι σημειώσεις, οι διαστάσεις και οι ανοχές μπορούν στη συνέχεια να προστεθούν εύκολα στο σχέδιο όπως απαιτείται. Η ενότητα σχεδίασης περιλαμβάνει τα περισσότερα μεγέθη και πρότυπα χαρτιού όπως *ANSI , ISO , DIN , GOST , JIS , BSI* και *SAC*.

Κυρίως σώμα

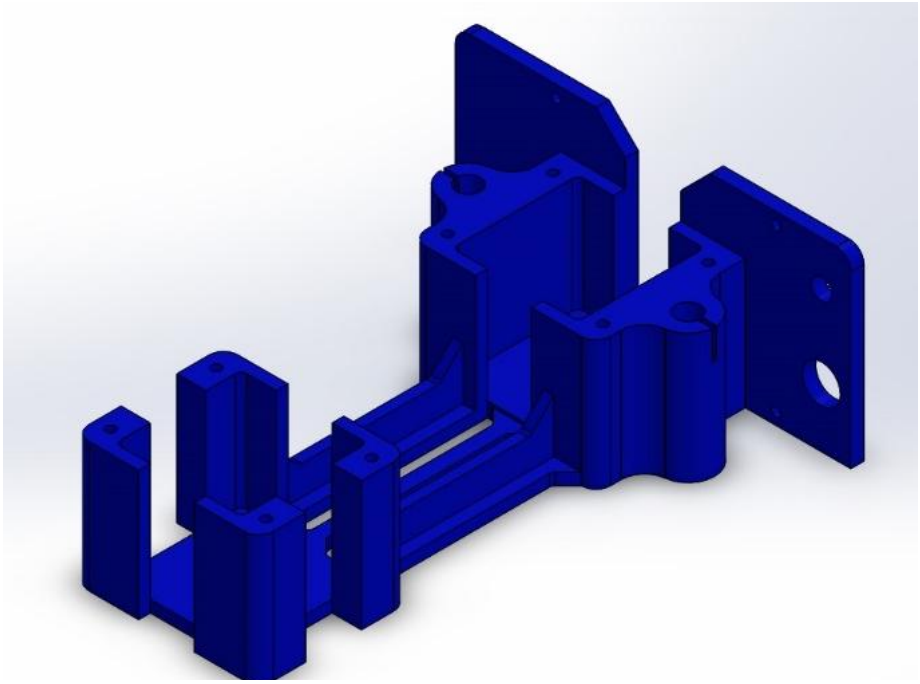
Το κυρίως σώμα είναι το εξάρτημα στο οποίο εδράζουν όλα τα εξαρτήματα και παρέχει την κατάλληλη αντοχή και στιβαρότητα στην συσκευή. Επάνω σε αυτό στηρίζονται οι βηματικοί κινητήρες περιστρεφόμενης τράπεζας και ανύψωσης του αισθητήρα, οι άξονες ολίσθησης ανύψωσης του αισθητήρα, και τα ηλεκτρονικά στοιχεία όπως η πλακέτα και τα πλήκτρα ελέγχου της συσκευής. Για το σχεδιασμό του κυρίως σώματος βασική προϋπόθεση είναι η γνώση των παρακάτω διαστάσεων:

- Διαστάσεις βηματικού ηλεκτροκινητήρα: Μπορεί να γίνει μέτρηση με το παχύμετρο ή να βρεθούν από το κατασκευαστικό σχέδιο που είναι αναρτημένο στο διαδίκτυο για κάθε τύπο κινητήρα. Στην συγκεκριμένη περίπτωση έχει επιλεχθεί ο βηματικός κινητήρας με διαστάσεις (42,3 x 42,3 x 40)mm.
- Απόσταση αισθητήρα *TOF* από την περιστρεφόμενη τράπεζα: Σε προηγούμενο κεφάλαιο επιλέχθηκε ο αισθητήρας απόστασης *TOF VL830X*. Στο τεχνικό φυλλάδιο αναγράφεται ότι το εύρος μέτρησης του συγκεκριμένου αισθητήρα είναι από 2mm έως 100mm. Επομένως οι βηματικοί κινητήρες θα τοποθετηθούν σε τέτοια απόσταση μεταξύ τους έτσι ώστε το κέντρο της περιστρεφόμενης τράπεζας από τον αισθητήρα να είναι 100mm.
- Έκκεντρη απόσταση των αξόνων ολίσθησης: Στο κυρίως σώμα στηρίζονται και οι κατακόρυφοι άξονες όπου ολισθαίνουν τα γραμμικά ρουλεμάν μαζί με το εξάρτημα στήριξης του αισθητήρα απόστασης. Η μεταξύ τους απόσταση

πρέπει να είναι τέτοια έτσι ώστε το εξάρτημα στήριξης που ολισθαίνει να χωράει τα γραμμικά ρουλεμάν, το περικόχλιο και τον αισθητήρα απόστασης.

- Μέγεθος και σημεία έδρασης της ηλεκτρονικής πλακέτας: Η ηλεκτρονική πλακέτα πρέπει να τοποθετηθεί σε σημείο εύκολα προσβάσιμο από τον χρήστη και σε τέτοια θέση σε σχέση με τα πλήκτρα και τους διακόπτες ώστε να είναι εύκολη η καλωδίωση τους. Το μέγεθος της ηλεκτρονικής πλακέτας που έχει επιλεχθεί σε προηγούμενο κεφάλαιο είναι (60x80)mm και οι οπές στήριξης (58x78)mm.
- Διαστάσεις οπών στήριξης πλήκτρων και διακοπτών: Η συσκευή σάρωσης διαθέτει έναν διακόπτη *ON-OFF*, ένα θηλυκό βύσμα *JACK* σύνδεσης τροφοδοτικού καθώς και ένα πλήκτρο στιγμιαίας επαφής για την λειτουργία σάρωσης με κάρτα μνήμης.

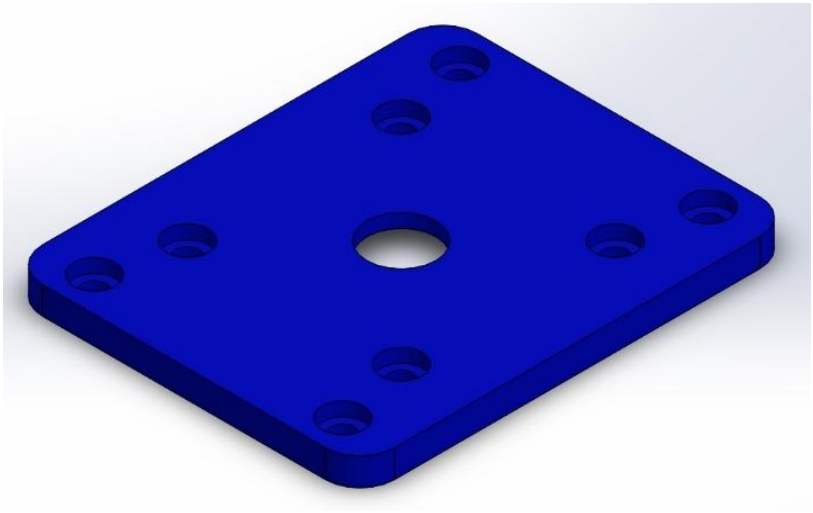
Στην παρακάτω **Εικόνα 5.1** παρουσιάζεται το ηλεκτρονικό τρισδιάστατο σχέδιο του κυρίου σώματος σε ισομετρική όψη:



Εικόνα 5.1: Ηλεκτρονικό σχέδιο κυρίους σώματος σαρωτή.

Στήριξη βηματικών κινητήρων

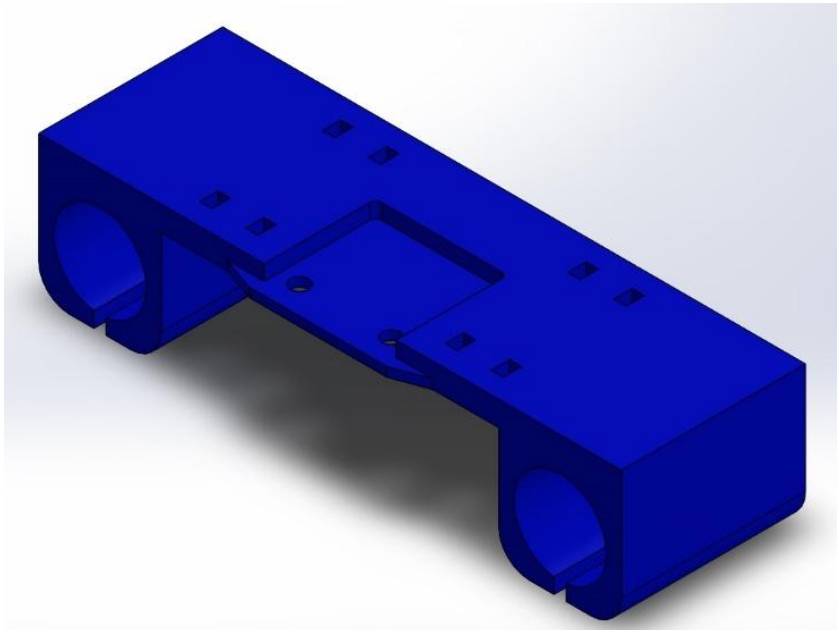
Οι βηματικοί κινητήρες φωλιάζουν στο κυρίως σώμα αλλά δεν συγκρατούνται από κάπου. Για την στερεή συναρμογή τους σχεδιάζεται μία πλάκα συγκράτησης η οποία βιδώνεται επάνω στους κινητήρες και στο κυρίως σώμα. Οι κρίσιμες διαστάσεις για τον σχεδιασμό αυτών είναι η απόσταση των οπών να ταιριάξουν με τις οπές του βηματικού όπου θα βιδώνονται οι βίδες. Από τα κατασκευαστικά σχέδια του βηματικού κινητήρα οι αποστάσεις των οπών είναι 32,5mm. Στην παρακάτω **Εικόνα 5.2** φαίνεται σε ισομετρική όψη το ηλεκτρονικό σχέδιο των 2 πλακών στήριξης:



Εικόνα 5.2: Ηλεκτρονικό σχέδιο πλάκας συγκράτησης βηματικού κινητήρα.

Βάση αισθητήρα TOF

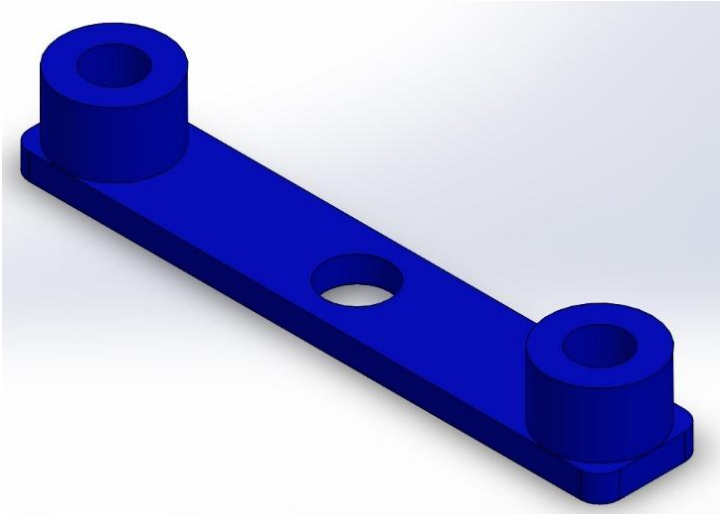
Ο αισθητήρας απόστασης θα πρέπει να κινείται κατά το ύψος του αντικειμένου στον άξονα Z έτσι ώστε να είναι δυνατή η συνολική σάρωση της επιφάνειας. Για την επίτευξη της γραμμικής κίνησης συνδέεται στον βηματικό κινητήρα ένας κοχλίας τραπεζοειδούς σπειρώματος και στην βάση στήριξης του αισθητήρα ένα περικόχλιο. Με αυτό τον τρόπο η περιστροφική κίνηση του κοχλία μετατρέπεται σε γραμμική του περικοχλίου άρα και του αισθητήρα. Δύο γραμμικά ρουλεμάν εξωτερικής διαμέτρου 15mm είναι σφηνωμένα αριστερά και δεξιά του αισθητήρα και ολισθαίνουν σε άξονες σκληρυμένου χάλυβα 8mm διατηρώντας έτσι την στιβαρότητα και την ευθυγράμμιση του κινούμενου μέρους. Η ισομετρική όψη του ψηφιακού σχεδίου παρουσιάζεται στην παρακάτω **Εικόνα 5.3**:



Εικόνα 5.3: Ηλεκτρονικό σχέδιο βάσης ολίσθησης του αισθητήρα.

Άνω συγκράτηση αξόνων ολίσθησης

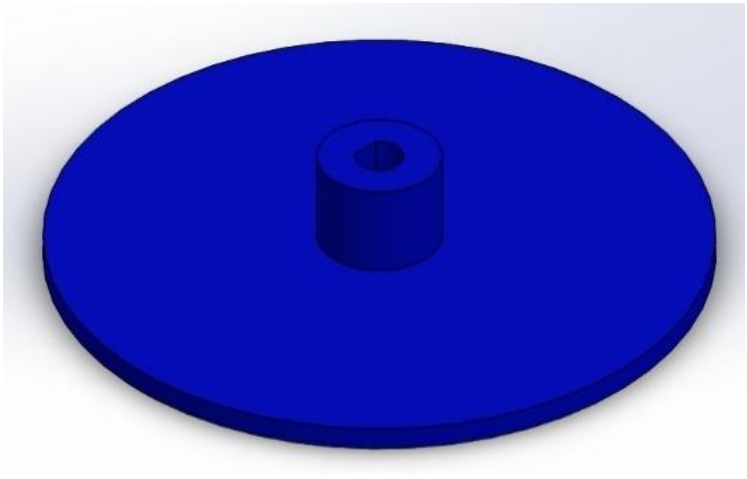
Οι δύο άξονες καταλήγουν στο άνω άκρο σε ένα πλαστικό εξάρτημα και σφηνώνουν ώστε η κατασκευή να γίνει πιο σταθερή. Να σημειωθεί ότι ο κοχλίας αφήνεται ελεύθερος χωρίς να συγκρατείται από ρουλεμάν. Αυτό γίνεται έτσι ώστε να κινείται ελεύθερος εντός των ανοχών του περικοχλίου και λόγω διαστατικών ατελειών τις κατασκευής τους χωρίς να επιβαρύνεται η συνολική κατασκευή με δυνάμεις. Παρουσιάζεται η ισομετρική όψη στην **Εικόνα 5.4**:



Εικόνα 5.4: Ηλεκτρονικό σχέδιο άνω συγκράτησης αξόνων.

Περιστρεφόμενη τράπεζα

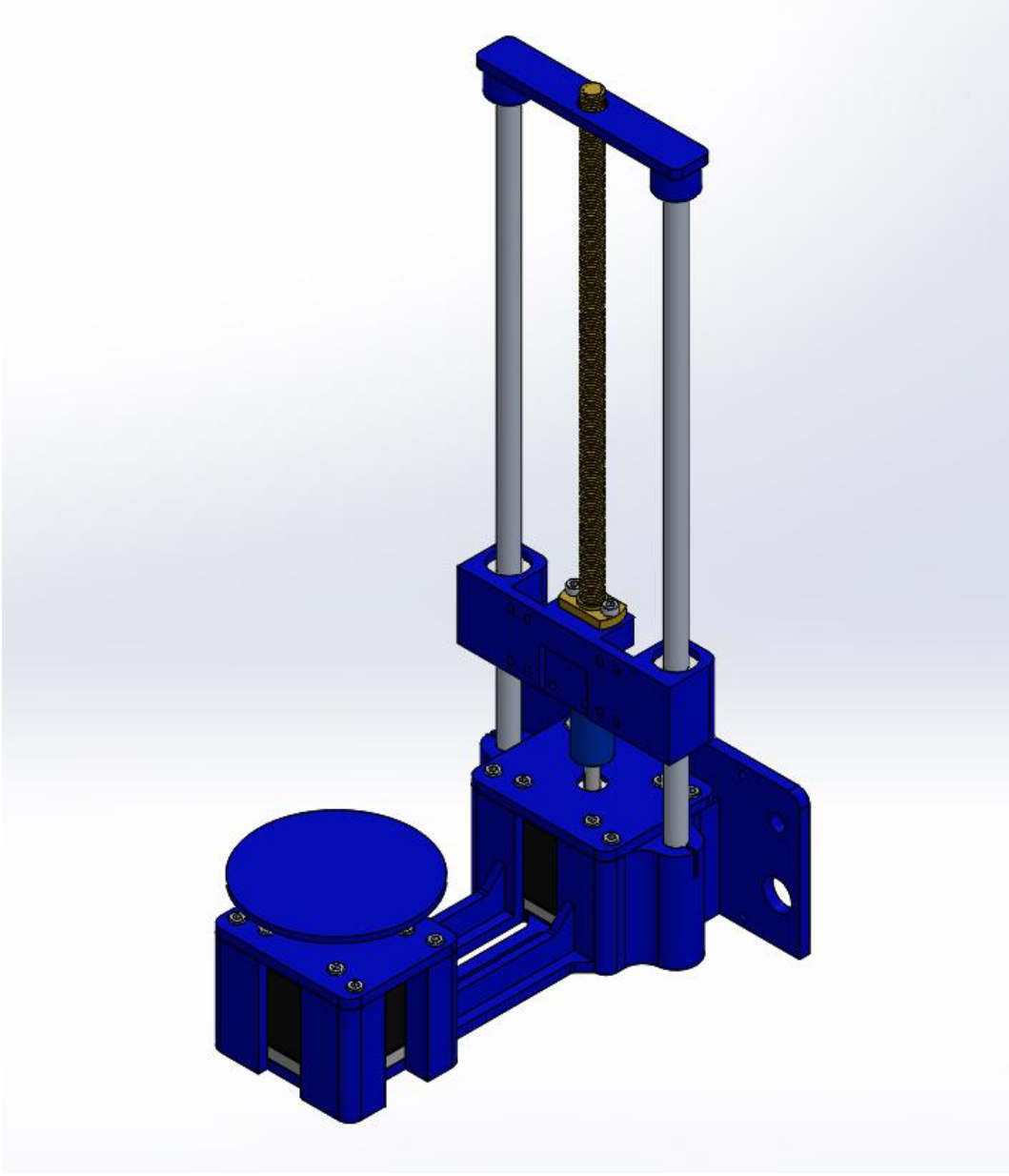
Σχεδιάζεται περιστρεφόμενη τράπεζα η οποία σφηνώνει στον άξονα του βηματικού κινητήρα. Να σημειωθεί ότι στην περιστρεφόμενη τράπεζα μπορεί να στηριχθεί αντικείμενο μεγαλύτερης διαμέτρου από αυτή. Στην **Εικόνα 5.5** φαίνεται σε ψηφιακή μορφή:



Εικόνα 5.5: Ηλεκτρονικό σχέδιο περιστρεφόμενης τράπεζας.

5.2 Έλεγχος συναρμογής

Το κυρίως σώμα εισάγεται ως το σταθερό εξάρτημα και πάνω σε αυτό συναρμολογούνται τα υπόλοιπα μέρη με σχέσεις εφαπτόμενων επιφανειών και ομόκεντρων οπών. Μπορούν επίσης να οριστούν περιστρεφόμενα μέρη ή να περιοριστεί η κίνηση ενός εξαρτήματος μεταξύ κάποιων ορίων. Στο συγκεκριμένο σχέδιο η τράπεζα μπορεί να περιστρέφεται και η βάση του αισθητήρα απόστασης να κινείται μεταξύ της αρχής σύνδεσης του κοχλία στον βηματικό κινητήρα έως και το πάνω μέρος στήριξης των αξόνων. Με αυτό το τρόπο είναι δυνατόν να ελεγθούν τυχόν διαστατικά λάθη μεταξύ των επιμέρους κομματιών της συναρμογής και εκκεντρότητες. Στην ηλεκτρονική συναρμολόγηση έχουν σχεδιαστεί επίσης οι βηματικοί κινητήρες, τα γραμμικά ρουλεμάν, ο κοχλίας με το περικόχλιο και οι άξονες ολίσθησης και η τελική μορφή φαίνεται στην παρακάτω **Εικόνα 5.6**.

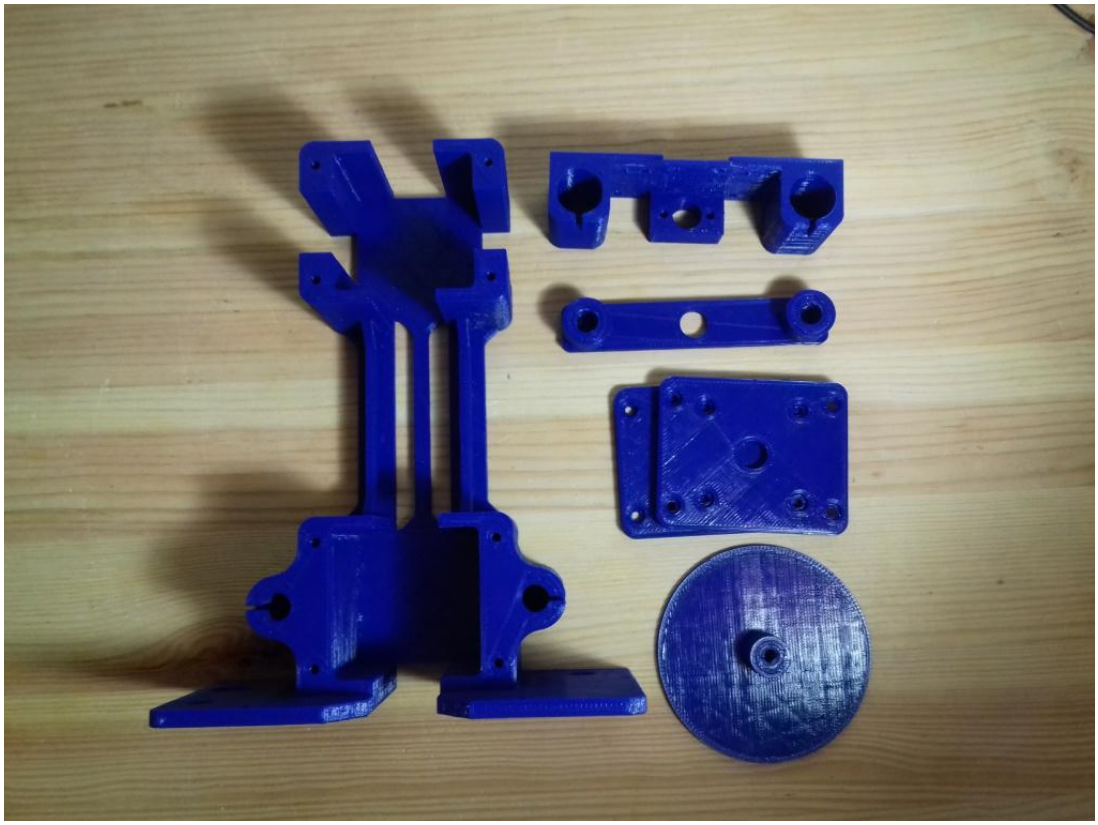


Εικόνα 5.6: Ηλεκτρονική συναρμογή του σαρωτή.

5.3 3D Εκτύπωση των εξαρτημάτων

Μετά τον σχεδιασμό και τον έλεγχο των εξαρτημάτων στην ηλεκτρονική συναρμογή, σειρά έχει αυτά να κατασκευαστούν και να συναρμολογηθούν. Θα χρησιμοποιηθεί τρισδιάστατος εκτυπωτής και τα κομμάτια θα εκτυπωθούν σε πλαστικό PLA. Η διαδικασία της εκτύπωσης περιλαμβάνει τα παρακάτω στάδια:

1. Αποθήκευση του ηλεκτρονικού σχεδίου σε μορφή αρχείου *STL*.
2. Εισαγωγή του αρχείου *STL* σε ένα λογισμικό *slicer* έτσι ώστε να υπολογιστεί το μονοπάτι εκτύπωσης εκφρασμένο σε κώδικα *G*. Είναι σημαντικό να έχει γίνει η σωστή παραμετροποίηση του *slicer* έτσι ώστε ο εκτυπωτής να τυπώνει στην επιθυμητή ταχύτητα και να εξωθεί την κατάλληλη ποσότητα λιωμένου πλαστικού καθώς και τις σωστές θερμοκρασίες της θερμαινόμενης τράπεζας και του ακροφύσιο ανάλογα με το υλικό.
3. Εισαγωγή του κώδικα *G* στον τρισδιάστατο εκτυπωτή, το μηχάνημα θα εκτελέσει τις απαραίτητες λειτουργίες και κινήσεις σύμφωνα με τις οδηγίες που παρέχονται συμβολικά στον κώδικα *G*. Να σημειωθεί ότι για την σωστή εκτύπωση του κομματιού θα πρέπει να έχει ευθυγραμμιστεί η τράπεζα με το ακροφύσιο έτσι ώστε σε οποιαδήποτε θέση να απέχει ίδια απόσταση. Η ποιότητα της εκτύπωσης εξαρτάται πάρα πολύ από την ποιότητα της πρώτης στρώσης πλαστικού. Τα εκτυπωμένα κομμάτια παρουσιάζονται στην παρακάτω **Εικόνα 5.7**:



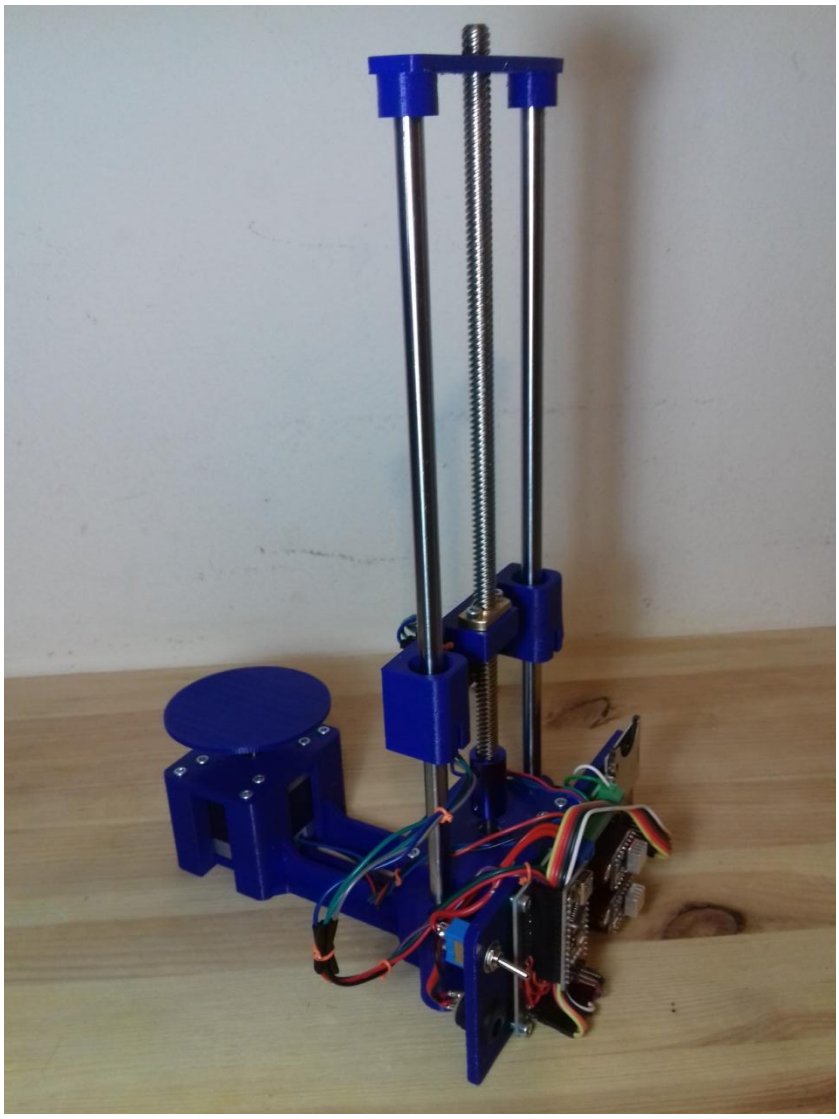
Εικόνα 5.7: Κομμάτια σαρωτή εκτυπωμένα σε τρισδιάστατο εκτυπωτή.

5.4 Τελική συναρμολόγηση σαρωτή

Αφού έχουν σχεδιαστεί και υλοποιηθεί όλα τα μηχανικά και ηλεκτρονικά στοιχεία της συσκευής μένει αυτά να συναρμολογηθούν. Η διαδικασία συναρμολόγησης του τρισδιάστατου σαρωτή περιλαμβάνει:

1. Τοποθέτηση των βηματικών κινητήρων στις υποδοχές τις μεγάλης βάσης.
2. Στήριξη των βηματικών κινητήρων με το βίδωμα των πλακών επάνω στο σώμα και στους ίδιους.
3. Τοποθέτηση των οδηγών αξόνων στις υποδοχές της κύριας βάσης.
4. Εφαρμογή των γραμμικών ρουλεμάν στις υποδοχές επάνω στην βάση στήριξης του αισθητήρα.
5. Εφαρμογή και βίδωμα του περικοχλίου στην κεντρική οπή της βάσης στήριξης του αισθητήρα.
6. Στήριξη του κοχλία του στον άξονα του βηματικού με τον κατάλληλο σύνδεσμο.
7. Βίδωμα του κοχλία με το περικόχλιο και ευθυγράμμιση της βάσης του αισθητήρα, των γραμμικών ρουλεμάν με τους άξονες.
8. Αφού περάσει η βάση στους άξονες, τοποθέτηση άνω στηρίγματος των αξόνων ευθυγράμμισης για περαιτέρω στιβαρότητα του κατακόρυφου άξονα κίνησης.
9. Τοποθέτηση τράπεζας στον άξονα του άλλου βηματικού κινητήρα.
10. Βίδωμα της ηλεκτρονικής πλακέτα, της SD card breakout, του διακόπτη τροφοδοσίας, του power jack και του πλήκτρου έναρξης σάρωσης επάνω στο πίσω μέρος της κύριας βάσης στις κατάλληλες σχεδιασμένες υποδοχές.
11. Στήριξη με βίδες του αισθητήρα απόστασης στην κινούμενη κατά τον άξονα Z βάσης στήριξης.
12. Καλωδίωση του ηλεκτρονικού κυκλώματος με τον αισθητήρα, τους βηματικούς κινητήρες, την τροφοδοσία και το SD card breakout σύμφωνα με το ηλεκτρονικό σχηματικό.

Η ολοκληρωμένη κατασκευή μετά την συναρμολόγηση παρουσιάζεται στην παρακάτω **Εικόνα 5.8**:



Εικόνα 5.8: Ολοκληρωμένη κατασκευή σαρωτή.

6. ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

6.1 Κώδικας Arduino

Στους μικροελεγκτές ο πηγαίος κώδικας ο οποίος διαχειρίζεται τις φυσικές εισόδους-εξόδους του ελεγκτή και εκτελεί τις απαραίτητες λειτουργίες και υπολογισμούς συνήθως ανεβαίνει στην πλακέτα μέσω ενός εξωτερικού προγραμματιστή.

Η σύνταξη του κώδικα για το *Arduino Nano* στα πλαίσια της πτυχιακής συντάσσεται στο περιβάλλον *Arduino IDE* και σε γλώσσα προγραμματισμού *C++ Wiring*. Πρόκειται για μία παραλλαγή της μητρικής γλώσσας *C++* κατά την οποία ο μικροελεγκτής μπορεί να προγραμματιστεί έτσι ώστε να παράγει παλμούς, να διαβάζει σήματα, να μετράει συχνότητες και να εκτελεί τους απαραίτητους υπολογισμούς που απαιτεί η εκάστοτε εφαρμογή. Ο κώδικας μέσω του *Arduino IDE* αποσφαλματοποιείται και στην συνέχεια ανεβάζεται στην πλακέτα μέσω της θύρας *USB* που είναι συνδεδεμένος. Ο λόγος που δεν χρειάζεται εξωτερικός προγραμματιστής στην διαδικασία ανεβάσματος του κώδικα είναι επειδή έχει προηγηθεί εγγραφή ενός υλικό-λογισμικού που ονομάζεται «*bootloader*» από την κατασκευάστρια εταιρεία. Να σημειωθεί ότι η εγγραφή του *bootloader* στο *Arduino* πραγματοποιείται με χρήση παράλληλου εξωτερικού προγραμματιστή σε αρχικό στάδιο.

Στην προκειμένη περίπτωση του τρισδιάστατου σαρωτή, το *Arduino Nano* καλείται:

1. Να δέχεται ένα σήμα εισόδου έναρξης σάρωσης και αποθήκευσης σε κάρτα μνήμης από ένα πλήκτρο, ή μία εντολή έναρξης σάρωσης από την θύρα *USB*.
2. Να ελέγχει δύο βηματικούς κινητήρες κατάλληλα, της περιστρεφόμενης τράπεζας και της ανύψωσης του αισθητήρα.
3. Να διαβάζει τις μετρήσεις απόστασης από τον αισθητήρα και να τις αποθηκεύει στην κάρτα μνήμης ή να τις στέλνει μέσω της θύρας *USB*.

Πιο συγκεκριμένα η συσκευή περιμένει σε κατάσταση αναμονής χωρίς να εκτελεί κάποια λειτουργία, αλλά ελέγχοντας συνεχώς μία συνδεδεμένη είσοδο με το πλήκτρο έναρξης σάρωσης ή αν υπάρχει εισερχόμενη εντολή από την θύρα *USB*. Εφόσον υπάρξει κάποιο ερέθισμα τότε ξεκινάει την ρουτίνα σάρωσης και κίνησης των βηματικών κινητήρων. Στον βηματικό κινητήρα στέλνονται ο κατάλληλος αριθμός σημάτων έτσι ώστε να περιστραφεί κατά 0,9 μοίρες κάθε φορά. Στην συνέχεια κάνει μία μικρή παύση για να διαβάσει την απόσταση από τον αισθητήρα. Ανάλογα με την αρχική είσοδο αποθηκεύει στην κάρτα μνήμης ή στέλνει μέσω της θύρας *USB* την πληροφορία. Έπειτα ξαναπεριστρέφει την τράπεζα κατά 0,9 και η διαδικασία αυτή γίνεται έως ότου το αντικείμενο περιστραφεί κατά 360 μοίρες. Με την ολοκλήρωση μίας περιστροφής το *Arduino* στέλνει τον κατάλληλο αριθμό βημάτων σε σταθερή συχνότητα στον οδηγό του κινητήρα ανύψωσης τους αισθητήρα, έτσι ώστε αυτός να ανέβει κατά ένα χιλιοστό. Η διαδικασία της σάρωσης ολοκληρώνεται όταν το αντικείμενο σαρωθεί σε όλο το ύψος του. Μετά την διεκπεραίωση της διαδικασίας ο αισθητήρας επαναφέρεται αυτόματα ξανά στην αρχική θέση εκκίνησης.

Για την σύνταξη του κώδικα χρησιμοποιήθηκαν οι παρακάτω βιβλιοθήκες:

- *SPI.h:* Για την επικοινωνία με την *SD card breakout*
- *SdFat.h:* Η συγκεκριμένη βιβλιοθήκη χρησιμοποιείται για την εύκολη εγγραφή και ανάγνωση αρχείων στην κάρτα μνήμης.
- *Wire.h:* Διαχειρίζεται την επικοινωνία με τον αισθητήρα απόστασης.
- *Adafruit_VL6180X.h:* Χρησιμοποιείται για την ανάγνωση της απόστασης από τον αισθητήρα.

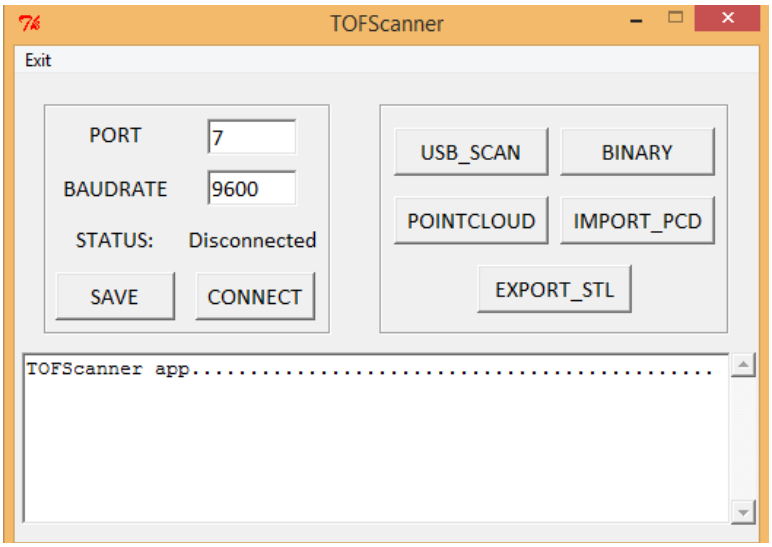
Ο ολοκληρωμένος κώδικας που διαχειρίζεται όλες τις λειτουργίες του σαρωτή και τρέχει στο *Arduino Nano* παρουσιάζεται στο ΠΑΡΑΡΤΗΜΑ Γ με τα κατάλληλα σχόλια για την εύκολη ανάγνωση του.

6.2 Κώδικας Python

Η *Python* είναι μια γλώσσα προγραμματισμού γενικής χρήσης υψηλού επιπέδου και δημιουργήθηκε από τον *Guido van Rossum* με πρώτο έτος κυκλοφορίας το 1991. Η συγκεκριμένη γλώσσα προγραμματισμού δίνει έμφαση στην ευκολία αναγνωσιμότητας του κώδικα με χρήση του σημαντικού κενού χώρου. Είναι μία γλώσσα προγραμματισμού πολλαπλών παραδειγμάτων και υποστηρίζει αντικειμενοστραφή αλλά και δομημένο προγραμματισμό. Στην συγκεκριμένη περίπτωση της πτυχιακής εργασίας η χρήση της *Python* γίνεται στα πλαίσια:

1. Να παρέχεται ένα φιλικό προς το χρήστη περιβάλλον με όλα τα απαραίτητα στοιχεία εισόδου εξόδου μηνυμάτων, εντολών και ελέγχου όπως: κουμπιά εντολών, κελία εισαγωγής ρυθμίσεων της λειτουργίας του σαρωτή και παράθυρο μηνυμάτων.
2. Να υπολογίζει και να εξάγει το σύννεφο σημείων της επιφάνειας του αντικειμένου προς σάρωση από τις μετρήσεις που παίρνει από τον αισθητήρα απόστασης είτε σε αποθηκευμένο δυαδικό αρχείο είτε από την θύρα *USB* σε πραγματικό χρόνο.
3. Να καλεί το πρόγραμμα ανακατασκευής επιφάνειας με δεδομένο το νέφος σημείων που είναι γραμμένο στην *MATLAB*.

Στην παρακάτω **Εικόνα 6.1** παρουσιάζεται το παράθυρο που δημιουργήθηκε με την βοήθεια του εργαλείου *Tkinter* της *Python*.



Εικόνα 6.1: Παράθυρο της εφαρμογής του σαρωτή.

Η εφαρμογή ονομάστηκε *TOFScanner* όπως φαίνεται και στο όνομα του παραθύρου. Παρακάτω αναλύονται οι λειτουργίες κάθε στοιχείου που αποτελεί το παράθυρο.

- Κουμπί *Exit* στην άνω μπάρα: Όταν πατιέται το συγκεκριμένο τότε η εφαρμογή κλείνει και τερματίζεται οποιαδήποτε διαδικασία.
- Κελί *PORT*: Αντιστοιχεί στον αριθμό της εικονικής σειριακής θύρας που φαίνεται ότι είναι συνδεδεμένο το *Arduino Nano* με *USB* στον υπολογιστή. Ο αριθμός αυτός φαίνεται στο περιβάλλον *Arduino IDE* στην καρτέλα *εργαλεία->θύρα* ή στην διαχείριση συσκευών του υπολογιστή.
- Κελί *BAUDRATE*: Όπως ειπώθηκε και σε προηγούμενα κεφάλαια είναι ο ρυθμός μετάδοσης των δεδομένων σε σειριακή ασύγχρονη επικοινωνία και πρέπει να είναι ο ίδιος και από την πλευρά του *Arduino Nano*.
- Ένδειξη *STATUS*: Η συγκεκριμένη ένδειξη παίρνει τις τιμές *Connected* ή *Disconnected* ανάλογα με το αν ο σαρωτής έχει συνδεθεί ή όχι με την εφαρμογή.
- Πλήκτρο *SAVE*: Με την πίεση αυτού του κουμπιού καλείται η υποσυνάρτηση *save_connect_settings()* στον κώδικα της *Python* η οποία διαβάζει από τα κελιά των ρυθμίσεων και αποθηκεύει τα δεδομένα σε ένα αρχείο κειμένου. Να σημειωθεί ότι κάθε φορά που η εφαρμογή τρέχει πρώτη φορά, διαβάζει τις αποθηκευμένες τιμές της τελευταίας φοράς και τις θέτει στα κελιά ώστε να γνωρίζει ο χρήστης τις τρέχουσες ρυθμίσεις. Επιπλέον σε κάθε αποθήκευση εμφανίζεται το μήνυμα «*New settings saved*» στο παράθυρο μηνυμάτων.
- Πλήκτρο *CONNECT*: Με το πάτημα του συγκεκριμένου καλείται η υποσυνάρτηση *connect()* του κώδικα η οποία διαβάζει τις αποθηκευμένες ρυθμίσεις στο αρχείο και προσπαθεί να συνδεθεί στο Arduino. Ο χαρακτήρας «*c*» στέλνεται μέσω της θύρας *USB* για να δηλώσει στον μικροελεγκτή ότι πρόκειται για εντολή σύνδεσης. Στην συνέχεια το *Arduino* στέλνει πίσω τον χαρακτήρα «*y*» και αν όλα πάνε καλά τότε η ένδειξη *STATUS* αλλάζει σε *Connected* καθώς και τυπώνεται το μήνυμα «*Successfully connected*». Αν για οποιοδήποτε λόγο δεν τηρηθεί η παραπάνω ακολουθία τότε σημαίνει ότι η εφαρμογή δεν έχει συνάψει επικοινωνία με τον σαρωτή οπότε το *STATUS*

μεταβάλλεται σε *Disconnected* και εμφανίζεται το ακόλουθο μήνυμα: «*Failed to connect*».

- Κουμπί *USB_SCAN*: Με το πάτημα του πλήκτρου καλείται η υποσυνάρτηση *usb_scan()* η οποία αρχικά ελέγχει αν υπάρχει σύνδεση με την συσκευή. Στην περίπτωση μη προηγηθείσας σύνδεσης ο χρήστης προειδοποιείται με το μήνυμα «*Try to connect first*». Αλλιώς καλείται την υπορουτίνα *usb_scan_process()* σε ξεχωριστό thread η οποία αρχικά ανοίγει ένα παράθυρο ώστε να επιλέξει ο χρήστης την διαδρομή όπου θα αποθηκευτεί το δυαδικό αρχείο. Έπειτα στέλνει τον χαρακτήρα «s» έτσι ώστε να δηλώσει την έναρξη της διαδικασίας σάρωσης μέσω *USB* ενώ παράλληλα τυπώνεται το μήνυμα «*USB scan request*» και ξεκινάει να διαβάζει τις μετρήσεις που στέλνει το Arduino και να τις αποθηκεύει σε ένα δυαδικό αρχείο και τυπώνεται το μήνυμα «*USB scan start*». Με το πέρας της σάρωσης τυπώνεται το μήνυμα «*USB scan completed*».
- Πλήκτρο *BINARY*: Με αυτό το κουμπί εισάγεται το δυαδικό αρχείο που δημιουργείται μετά από μία διαδικασία σάρωσης και στις δύο περιπτώσεις με αποθήκευση στην κάρτα μνήμης ή μέσω USB σύνδεσης και με την επιτυχή εισαγωγή τυπώνεται το μήνυμα «*Scan file imported*».
- Πλήκτρο *POINTCLOUD*: Αρχικά ζητάει από τον χρήστη να επιλέξει την επιθυμητή διαδρομή αποθήκευσης του σύννεφου σημείων και τυπώνει το μήνυμα «*Pointcloud calculation request*». Στην συνέχεια καλείται η υπορουτίνα *pointcloud_process()* που τυπώνει το μήνυμα «*Please wait this may take a while*» και η οποία διαβάζει τις αποθηκευμένες τιμές της απόστασης από δυαδικό αρχείο και υπολογίζει τις συντεταγμένες του σημείου επάνω στην επιφάνεια του αντικειμένου. Όλες οι συντεταγμένες αποθηκεύονται σε αρχείο κειμένου με την μορφή X;Y;Z και με την ολοκλήρωση υπολογισμού εμφανίζεται στον χρήστη το μήνυμα «*Pointcloud DONE*».
- Κουμπί *IMPORT_PLY*: Εισάγεται το επιθυμητό νέφος σημείων όπου θα ακολουθήσει ο υπολογισμός της επιφάνειας.
- Πλήκτρο *EXPORT_STL*: Καλεί τον αντικειμενικό κώδικα που έχει προέλθει από τον πηγαίο κώδικα της *MATLAB* για να υπολογίσει την επιφάνεια του αντικειμένου με βάση το νέφος σημείων και να αποθηκεύσει τα αποτελέσματα σε μορφή αρχείου *STL*.

Ο πηγαίος κώδικας της εφαρμογής γραμμένος σε *Python* βρίσκεται στο ΠΑΡΑΡΤΗΜΑ Γ.

6.3 Κώδικας MATLAB

Η *MATLAB* είναι ένα περιβάλλον αριθμητικής υπολογιστικής και μία γλώσσα προγραμματισμού τέταρτης γενιάς. Κύριες λειτουργίες που έχει την δυνατότητα να εκτελεί είναι η σχεδίαση συναρτήσεων, πράξεις μαθηματικών πινάκων, εφαρμογή αλγορίθμων, δημιουργία διεπαφών με τον χρήστη καθώς και διασύνδεση με προγράμματα σε διαφορετικές γλώσσες προγραμματισμού.

Αν και η *MATLAB* προορίζεται κυρίως για αριθμητικούς υπολογισμούς, μια προαιρετική εργαλειοθήκη χρησιμοποιεί τη συμβολική μηχανή *MuPAD* που επιτρέπει την πρόσβαση σε συμβολικές δυνατότητες υπολογιστών. Ένα επιπλέον πακέτο, το *Simulink* , προσθέτει γραφική προσομοίωση πολλαπλών τομέων και βασισμένο σε μοντέλα σχεδιασμού για δυναμικά και ενσωματωμένα συστήματα.

Στην παρούσα πτυχιακή ο αλγόριθμος που χτίστηκε στην *MATLAB* σε πειραματικό στάδιο επιχειρεί να δημιουργήσει την επιφάνεια του αντικειμένου με βάση το σύννεφο σημείων που έχει υπολογιστεί προγενέστερα και το αποτέλεσμα να αποθηκευτεί σε μορφή αρχείου *STL*. Πιο συγκεκριμένα η διαδικασία περιλαμβάνει τα παρακάτω:

- Ανάγνωση των αποθηκευμένων συντεταγμένων του σύννεφου σημείων, dlmread.
- Αφαίρεση μη έγκυρων σημείων, *removeInvalidPoints*.
- Μείωση θορύβου των σημείων, *pcdenoise*.
- Αφαίρεση διπλότυπων σημείων, *unique*.
- Δημιουργία επιφάνειας με τον αλγόριθμο *MyCrustOpen*.
- Αποθήκευση των παραγόμενων επιπέδων και ακμών σε αρχείο μορφής *STL, stlwrite*.

Ο ολοκληρωμένος πηγαίος κώδικας του *MATLAB* παρουσιάζεται στο ΠΑΡΑΡΤΗΜΑ Γ.

6.3 Εκτελέσιμο αρχείο

Όσο αναφορά το πρόγραμμα στην πλευρά του υπολογιστή αυτό αποτελείται από δύο μέρη, από τον κώδικα γραμμένο σε *Python* και τον κώδικα χτισμένο στην *MATLAB*. Επιπλέον οι αλγόριθμοι εκτελούνται αντίστοιχα στην μορφή του πηγαίου τους κώδικα και επομένως ο υπολογιστής θα πρέπει να έχει εγκατεστημένα τα αντίστοιχα προγράμματα σύνταξης *Python* και *MATLAB* καθώς και όλες τις βιβλιοθήκες που χρησιμοποιούνται. Στην πράξη αυτό δεν είναι καθόλου πρακτικό καθώς κάθε φορά που το πρόγραμμα θα εκτελείται σε διαφορετικό υπολογιστή θα πρέπει να γίνεται εγκατάσταση όλων των παραπάνω. Όπως είναι γνωστό τα περισσότερα προγράμματα βρίσκονται συνήθως στην διαδρομή «*C:\Program Files*» και μία συντόμευση της εφαρμογής βρίσκεται στην επιφάνεια εργασίας. Τα αρχεία τα οποία περιέχουν τις ρυθμίσεις και διάφορες πληροφορίες τα οποία υπόκεινται σε επεξεργασία δεν μπορούν να βρίσκονται στην ίδια διαδρομή καθώς δεν επιτρέπονται οι αλλαγές σε αρχεία στον δίσκο «*C:*». Κατάλληλη θέση για τα αρχεία στα οποία γίνονται αλλαγές έχει καθιερωθεί για όλα τα προγράμματα η διαδρομή «*C:\Users\Administrator\AppData\Local*».

Για να παραχθεί το εκτελέσιμο αρχείο από τον πηγαίο κώδικα της γλώσσας προγραμματισμού της Python χρειάζεται να εγκατασταθεί το εργαλείο της *Python*, «*cxFreeze*». Αυτό απαιτεί ένα ακόμη αρχείο *Python* με το όνομα «*setup*» που θα βρίσκεται στον ίδιο φάκελο με τον πηγαίο κώδικα της εφαρμογής. Από την γραμμή εντολών του υπολογιστή στην διαδρομή του φακέλου εκτελείται η εντολή «*python setup.py build*». Με αυτό τον τρόπο παράγεται το εκτελέσιμο αρχείο μέσω του πηγαίου κώδικα της Python το οποίο μπορεί να τρέξει σε οποιοδήποτε υπολογιστή χωρίς την εγκατάσταση προγραμμάτων και βιβλιοθηκών.

Ο πηγαίος κώδικας της *MATLAB* μπορεί να μεταγλωττιστεί πιο εύκολα από το ίδιο περιβάλλον της *MATLAB*. Αυτό έχει ως αποτέλεσμα την δημιουργία ενός εκτελέσιμου αρχείου το οποίο μεταφέρεται στον ίδιο φάκελο με αυτό της Python.

Παρόλα αυτά κάθε φορά που κάποιος θέλει να εγκαταστήσει το πρόγραμμα στον υπολογιστή του θα πρέπει να έχει διαθέσιμα τον φάκελο με τα εκτελέσιμα αρχεία και έναν φάκελο με όλα τα αρχεία που υπόκεινται σε επεξεργασία και είναι αποθηκευμένες οι ρυθμίσεις. Στην συνέχεια χειροκίνητα να μεταφέρει τα εκτελέσιμα

στην διαδρομή «*C:\Program Files*» δημιουργώντας παράλληλα μία συντόμευση στην επιφάνεια εργασίας και τα αρχεία των ρυθμίσεων στην διαδρομή «*C:\Users\Administrator\AppData\Local*». Στην πράξη αυτό δεν είναι καθόλου λειτουργικό και γι αυτό θα δημιουργηθεί ένα εκτελέσιμο αρχείο το οποίο θα εξάγει αυτόματα όλα τα απαραίτητα αρχεία και προγράμματα στην κατάλληλη θέση, το λεγόμενο «setup.exe» (εκτελέσιμο αρχείο εγκατάστασης) του προγράμματος.

Για την δημιουργία του αρχείου εγκατάστασης χρησιμοποιείται η εφαρμογή «*Inno Setup*» η οποία είναι ένα δωρεάν πρόγραμμα δημιουργίας εκτελέσιμων αρχείων εγκατάστασης για προγράμματα *Windows* από τους *Jordan Russell* και *Martijn Laan*. Παρουσιάστηκε για πρώτη φορά το 1997, το *Inno Setup* ανταγωνίζεται σήμερα και ξεπερνά ακόμη και πολλούς εμπορικούς εγκαταστάτες σε σύνολο χαρακτηριστικών και σταθερότητα. Μέσα από μία διαδικασία που ακολουθείται, στην βάση του το *Inno Setup* παράγει έναν κώδικα σε γλώσσα *Pascal* η οποία μεταγλωττίζεται και δημιουργείται το αρχείο εγκατάστασης «*setup.exe*». Ο κώδικας *Pascal* που παρήγαγε το *Inno Setup* παρουσιάζεται στο ΠΑΡΑΡΤΗΜΑ Γ.

7. ΔΙΑΔΙΚΑΣΙΑ ΣΑΡΩΣΗΣ

Για την επίδειξη σάρωσης της διάταξης που υλοποιήθηκε στην παρούσα πτυχιακή εργασία θα χρησιμοποιηθεί μία κεραμική κούπα με χερούλι όπως φαίνεται στην παρακάτω **Εικόνα 7.1**. Αρχικά τοποθετείται στην περιστρεφόμενη τράπεζα με ταινία διπλής όψης και ελέγχεται ότι ο αισθητήρας βρίσκεται στο κατώτερο σημείο. Στην συνέχεια ο σαρωτής συνδέεται στην πρίζα και μπορεί να ενεργοποιηθεί από τον διακόπτη σύνδεσης και αποσύνδεσης της τροφοδοσίας.

Εικόνα 7.1: Αντικείμενο προς σάρωση

7.1 Σάρωση με κάρτα μνήμης

Η διαδικασία σάρωσης με αποθήκευση σε κάρτα μνήμης είναι πολύ χρήσιμη στις περιπτώσεις όπου η συσκευή δεν μπορεί να συνδεθεί άμεσα με καλώδιο *Mini-B USB* σε έναν υπολογιστή με εγκατεστημένη την εφαρμογή. Προαπαιτείται η κάρτα μνήμης να έχει υποστεί διαμόρφωση σε σύστημα αρχείων *FAT32*. Για να γίνει αυτό συνδέεται το αποθηκευτικό μέσο σε ένα υπολογιστή και όταν αυτό εμφανιστεί με δεξί κλικ εμφανίζεται το «*format*» από όπου και επιλέγεται το σύστημα αρχείων *FAT32*. Αφού γίνει η συγκεκριμένη προεργασία η οποία απαιτείται μόνο μία φορά τότε η κάρτα μνήμης εισάγεται στην υποδοχή του *SD card breakout* επάνω στον σαρωτή. Στην συνέχεια πιέζεται το πλήκτρο έναρξης σάρωσης και η διαδικασία ξεκινάει αποθηκεύοντας στην κάρτα μνήμης τις μετρήσεις του αισθητήρα σε ένα δυαδικό αρχείο με όνομα «*dataSD»*. Να σημειωθεί ότι η λειτουργία σάρωσης εμποδίζεται από

το ίδιο το πρόγραμμα στην περίπτωση όπου δεν έχει τοποθετηθεί η κάρτα μνήμης στην υποδοχή η συμβαίνει κάποιο σφάλμα εγγραφής του αρχείου.

7.2 Σάρωση μέσω USB

Η διαδικασία σάρωσης μέσω *USB* καλωδίου είναι χρήσιμη στην περίπτωση όπου δεν διαθέτεται μία κάρτα μνήμης. Απαιτείται μόνο ένα καλώδιο «Mini-B USB» συνδεδεμένο με ένα υπολογιστή όπου έχει εγκατασταθεί το λογισμικό *TOFScanner*. Πριν ξεκινήσει η σάρωση του αντικειμένου θα πρέπει να γίνει σύνδεση της συσκευής με το πρόγραμμα. Στο κελί *PORT* συμπληρώνεται ο αριθμός της σειριακής θύρας όπως αυτή εμφανίζεται στην διαχείριση συσκευών του υπολογιστή η πιο εύκολα στο *Arduino IDE* στην καρτέλα *Tools->Port*. Αφού συμπληρωθούν οι κατάλληλες παράμετροι αποθηκεύονται με το κουμπί *SAVE* και στην συνέχεια πατιέται το κουμπί *CONNECT*. Η συσκευή έχει συνδεθεί όταν η ένδειξη *STATUS* αλλάξει σε *Connected*. Η έναρξη της σάρωσης πυροδοτείται με το πάτημα του κουμπιού *USB_SCAN* όπου επιλέγεται και η διαδρομή αποθήκευσης σάρωσης του δυαδικού αρχείου που περιέχει τις μετρήσεις και δημιουργείται ένα δυαδικό αρχείο με το όνομα «*dataUSB*».

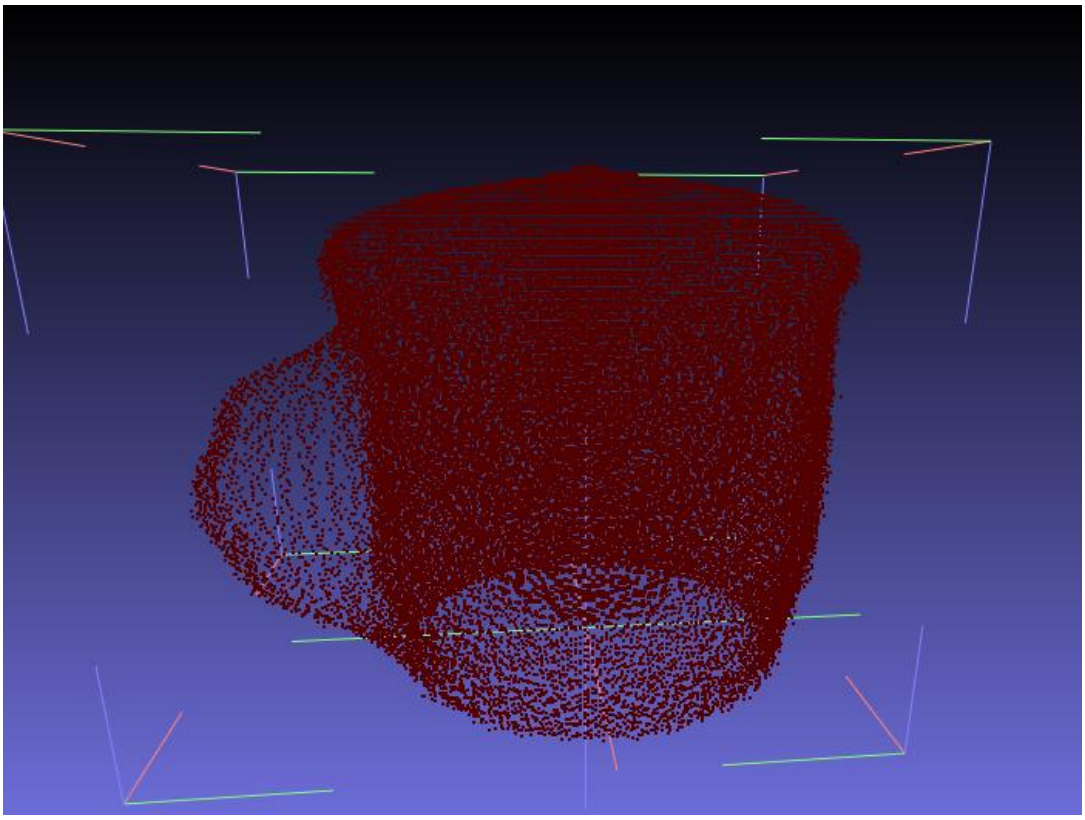
7.3 Παραγωγή νέφους σημείων

Η δειγματοληπτική σάρωση των σημείων της επιφάνειας του αντικειμένου και στις δύο παραπάνω περιπτώσεις έχει ως αποτέλεσμα την δημιουργία ενός δυαδικού αρχείου. Ο υπολογισμός των συντεταγμένων κάθε σημείου σε καρτεσιανό σύστημα πραγματοποιείται σε δεύτερο στάδιο. Πατώντας το κουμπί *Binary* εισάγεται το επιθυμητό δυαδικό αρχείο και στην συνέχεια με το κουμπί *POINTCLOUD* επιλέγεται η διαδρομή αποθήκευσης του αρχείου κειμένου που περιέχει τις συντεταγμένες των σημείων του νέφους. Η διαδικασία υπολογισμού ολοκληρώνεται με την εμφάνιση του μηνύματος «*Pointcloud completed*». Το παραγόμενο αρχείο είναι ένα αρχείο κειμένου με το όνομα «*pointcloud.txt*» το οποίο περιέχει της συντεταγμένες των σημείων του νέφους στην μορφή X;Y;Z.

7.4 Λογισμικό MeshLab

Το *MeshLab* είναι ένα σύστημα λογισμικού επεξεργασίας τρισδιάστατων πλεγμάτων και νέφους σημείων και παρέχει ένα σύνολο εργαλείων για την επεξεργασία, τον καθαρισμό, την επούλωση, τον έλεγχο, την απόδοση και τη μετατροπή αυτών. Πρόκειται για ένα δωρεάν και ανοιχτού κώδικα πρόγραμμα.

Μπορεί να χρησιμοποιηθεί για την επεξεργασία και το φιλτράρισμα του νέφους καθώς παρέχει πολλά περισσότερα εργαλεία επεξεργασίας του νέφους καθώς και αλγόριθμους υπολογισμού των επιφανειών. Στην **Εικόνα 7.2** φαίνεται το σύννεφο σημείων της κούπας σε αρχικό στάδιο.

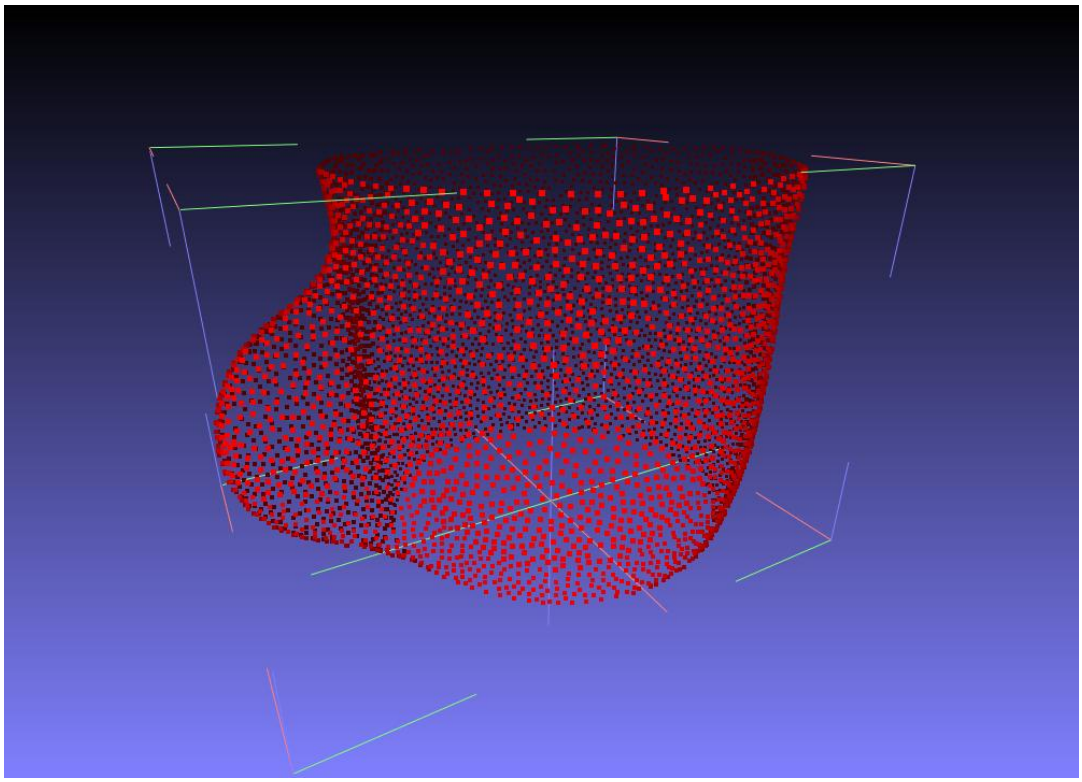


Εικόνα 7.2: Αρχικό νέφος σημείων στο MeshLab.

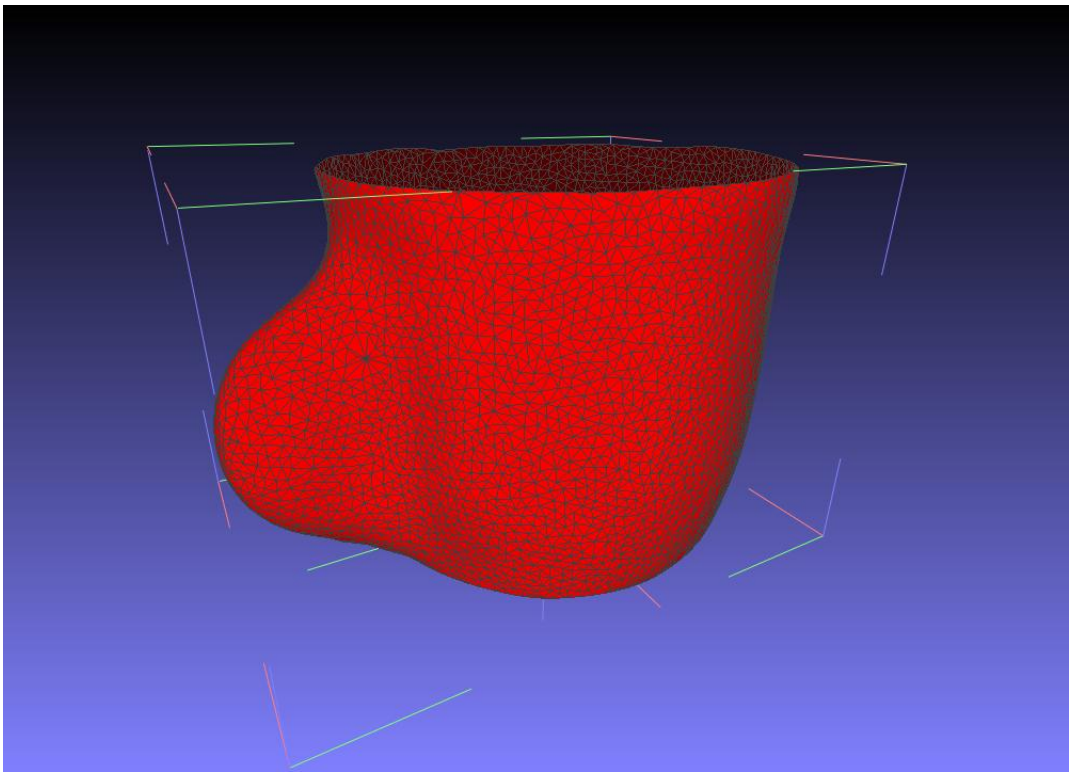
Η διαδικασία που ακολουθείται για το φιλτράρισμα και την εξομάλυνση του νέφους στο MeshLab περιλαμβάνει τα παρακάτω βήματα:

1. Εισαγωγή του νέφους, επιλέγεται *File->Import Mesh.*
2. Χειροκίνητη επιλογή και διαγραφή άκυρων σημείων.
3. *Filters->Point set->Compute normals for point sets.*
4. *Filters->Remeshing->Ball Pivoting*
5. *Filters->Smoothing->Laplacian smooth*
6. *Filters->Sampling->Poison disk sampling*
7. *Filters->Point set->Compute normals for point sets.*
8. *Filters->Remeshing->Ball Pivoting*

Το φιλτραρισμένο νέφος σημείων παρουσιάζεται στην **Εικόνα 7.3** καθώς και η ανακατασκευασμένη επιφάνεια του αντικειμένου στην **Εικόνα 7.4**.



Εικόνα 7.3: Φιλτραρισμένο νέφος σημείων στο MeshLab.



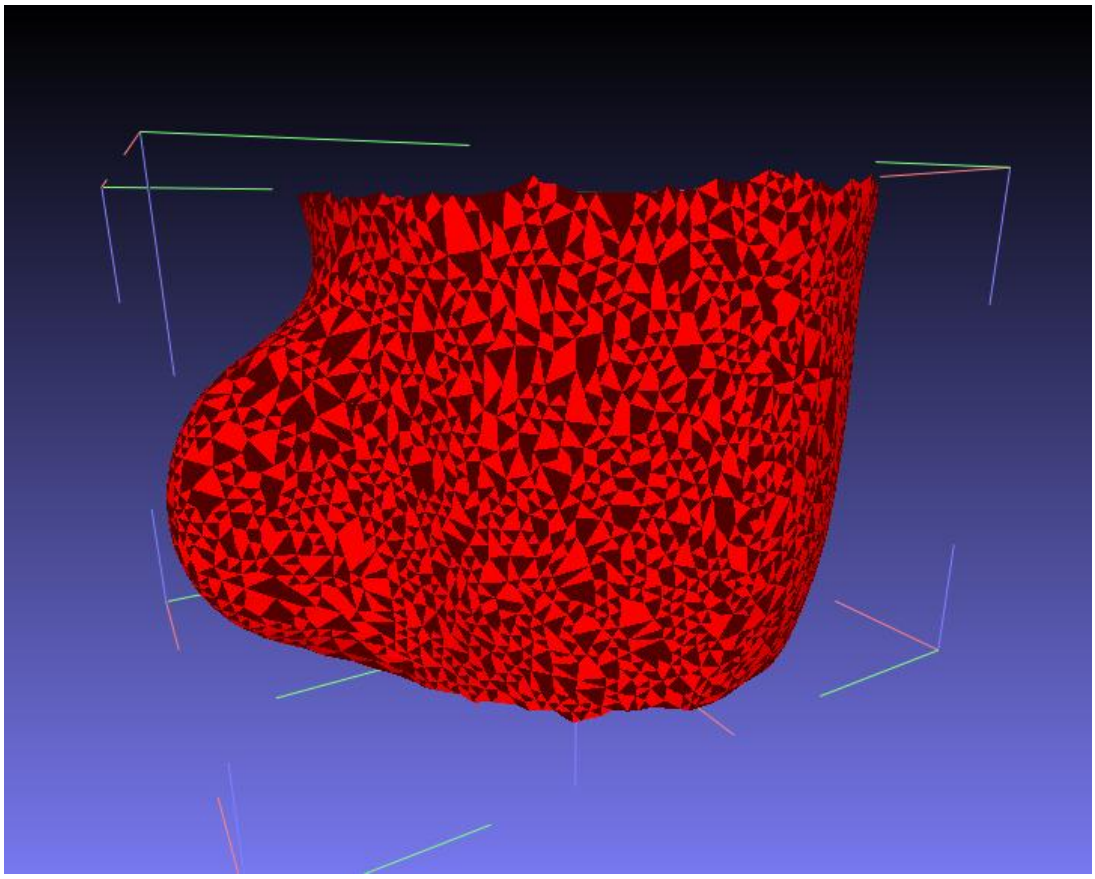
Εικόνα 7.4: Ανακατασκευασμένη επιφάνεια στο MeshLab με τον αλγόριθμο Ball Pivoting.

7.6 Δημιουργία STL αρχείου

Αφού πραγματοποιήθηκε διόρθωση, φιλτράρισμα και εξομάλυνση στο λογισμικό MeshLab, είναι δυνατόν το νέο νέφος σημείων να αποθηκευτεί σε αρχείο pointcloud και να εισαχθεί στο λογισμικό TOFScanner που υλοποιήθηκε στα πλαίσια της πτυχιακής έτσι ώστε να γίνει ανακατασκευή της επιφάνειας και αποθήκευση σε αρχείο STL από τον ενσωματωμένο αλγόριθμο MATLAB. Η διαδικασία περιλαμβάνει:

1. Εισαγωγή αρχείου νέφους σημείων με το κουμπί *IMPORT_PCD*
2. Ανακατασκευή επιφάνειας και αποθήκευση σε STL με το κουμπί *EXPORT_STL*.

Το αρχείο που δημιουργήθηκε από την αλγόριθμο της MATLAB εισάγεται στο MeshLab για την οπτικοποίηση του και φαίνεται στην παρακάτω **Εικόνα 7.5**.



Εικόνα 7.5: Ανακατασκευασμένη επιφάνεια από τον αλγόριθμο MATLAB όπως φαίνεται στο MeshLab.

8. ΣΥΜΠΕΡΑΣΜΑΤΑ

8.1 Συνολικό κόστος

Στον **πίνακα 8-1** υπολογίζεται το συνολικό κόστος κατασκευής αθροίζοντας όλα τα επιμέρους κόστη των εξαρτημάτων που αποτελείται χωρίς να περιλαμβάνονται τα κόστη για τα εργαλεία και τα αναλώσιμα υλικά που χρησιμοποιήθηκαν για την κατασκευή της διάταξης.

Υπολογισμός κόστους πλαστικών μερών

Τα πλαστικά αντικείμενα εκτυπώθηκαν στον προσωπικό μου τρισδιάστατο εκτυπωτή και το κόστος αγοράς του υλικού ανέρχεται στα 24 ευρώ ανά κιλό. Συνολικά όλα τα κομμάτια ζυγίστηκαν και βρέθηκαν να είναι 285 γραμμάρια. Με την απλή μέθοδο των τριών υπολογίζεται ότι το συνολικό κόστος του υλικού εκτύπωσης είναι 6,84 ευρώ.

Πίνακας 8-1: Συνολικό κόστος κατασκευής

ΕΞΑΡΤΗΜΑ	ΠΟΣΟΤΗΤΑ	ΤΙΜΗ ΑΝΑ ΤΕΜ.	ΚΟΣΤΟΣ
Arduino Nano clone	1	6,20	6,20
Τροφοδοτικό 12V	1	3,05	3,05
Βηματικός κινητήρας	2	14,89	29,78
DRV8825	2	4,40	8,80
SD Breakout	1	1,90	1,90
Adafruit VL6180X	1	17,40	17,40
Διάτρητη πλακέτα	1	1,50	1,5
Power jack 5.5X2.1mm	1	0,50	0,50
Momentary Button	1	0,50	0,50
1x40PIN 2,54mm Female	1	0,25	0,25
1x40PIN 2,54mm Male	1	0,25	0,25
Screw terminal 2P 2,54mm	1	0,40	0,40
Screw terminal 8P 2,54mm	1	1,80	1,80
KF301-2P 5.08mm terminal	1	0,25	0,25
Αντίσταση 10KOhm 2W	1	0,02	0,02
Πυκνωτές 50V 100uF	2	0,07	0,14
Καλώδια	1	2,00	2,00
Περικόχλιο T8	1	1,60	1,60
Κοχλίας T8 x 300mm	1	5,90	5,90
Άξονας D8mm x L300	2	3,00	6,00
Διακόπτης τροφοδοσίας 3A	1	0,60	0,60
Σύνδεσμος άξονα	1	1,80	1,80
Βίδα M3 L10 DIN912	26	0,10	2,60
Γραμμικό ρουλεμάν	2	1,60	1,60
Πλαστικά μέρη	-	6,84	6,84
ΣΥΝΟΛΙΚΟ ΚΟΣΤΟΣ ΚΑΤΑΣΚΕΥΗΣ			**103,28€**

8.2 Βελτιώσεις

Η πιο σημαντική βελτίωση που μπορεί να πραγματοποιηθεί στην διάταξη σάρωσης που υλοποιήθηκε στα πλαίσια της πτυχιακής εργασίας είναι η αναβάθμιση του αισθητήρα απόστασης. Ο κώνος αίσθησης του συγκεκριμένου είναι 25 μοίρες άνοιγμα και ελάχιστη υποδιαίρεση μέτρησης 1mm που έχει ως αποτέλεσμα ένα θορυβώδες νέφος σημείων το οποίο χρειάζεται φιλτράρισμα και εξομάλυνση, καθώς και οι λεπτομέρειες τις επιφάνειας του αντικειμένου μικρότερες από τον κώνο αίσθησης δεν εμφανίζονται πάντα σωστά.

Όσο αναφορά το προγραμματιστικό κομμάτι μπορούν προστεθούν περισσότερα εργαλεία επεξεργασίας και οπτικοποίησης του νέφους στον κώδικα Python καθώς και να εισαχθούν στον κώδικα MATLAB περισσότεροι αλγόριθμοι ανακατασκευής επιφανειών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] A. Alqudah, Survey of Surface Reconstruction Algorithms, *Journal of Signal and Information Processing*, 5, 63-79, 2014.
doi: 10.4236/jsip.2014.53009

[2] R.A. Jarvis, On the identification of the convex hull of a finite set of points in the plane*, The Australian National University,* 18-21, March 1973.
https://doi.org/10.1016/0020-0190(73)90020-3

[3] D.R. Chand, S.S. Kapur, An Algorithm for Convex Polytopes, *Journal of the ACM*, January 1970.
https://doi.org/10.1145/321556.321564

[4] J.S. Greenfield, A Proof for a QuickHull Algorithm *Electrical Engineering and Computer Science - Technical Reports* 65, April 1990.
https://surface.syr.edu/eecs_techreports/65/

[5] C.B. Barber, D.P. Dobkin, H. Huhdanpaa, The quickhull algorithm for convex hulls, *ACM Transactions on Mathematical Software*, December 1996.
https://doi.org/10.1145/235815.235821

[6] R.L Graham, An efficient algorithm for determining the convex hull of a finite planar set, *Bell Telephone Laboratories*, June 1972.
https://doi.org/10.1016/0020-0190(72)90045-2

[7] D. Freedman, An incremental algorithm for reconstruction of surfaces of arbitrary codimension, *Rensselaer Polytechnic Institute*, February 2007.
https://doi.org/10.1016/j.comgeo.2006.05.004

[8] Yuan-Zhi Lyu, 3D Point Cloud Surface Reconstruction Based on Divide-and-Conquer Method in Laser Scanner, *Journal of Physics*, March 2020.
https://iopscience.iop.org/article/10.1088/1742-6596/1544/1/012118/meta

[9] A. Bowyer, Computing Dirichlet tessellations, *School of Mathematics*, January 1981.
https://doi.org/10.1093/comjnl/24.2.162

[10] D. F. Watson, Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes, *Department of Geology and Geophysics*, January 1981.
https://doi.org/10.1093/comjnl/24.2.167

[11] J.D. Boissonnat, Geometric structures for three-dimensional shape representation, *ACM Transactions on Graphics*, October 1984.
https://doi.org/10.1145/357346.357349

[12] J. L. Bentley, J. H. Friedman, Data Structures for Range Searching, *ACM Computing Surveys,* December 1979.
https://doi.org/10.1145/356789.356797

[13] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Surface reconstruction from unorganized points, *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, July 1992.
https://doi.org/10.1145/133994.134011

[14] N. Amenta, M. Bern, M. Kamvysselis, A new Voronoi-based surface reconstruction algorithm*, Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, July 1998.
https://doi.org/10.1145/280814.280947

[15] Γ. Παντής, Πτυχιακή εργασία: Βηματικοί κινητήρες, κατασκευή και μεθοδολογία ελέγχου, Ηλεκτρολόγων Μηχανικών Τ.Ε., Πάτρα 2017.
http://repository.library.teimes.gr/xmlui/handle/123456789/5631

Διαδικτυακές πηγές

[δ1]https://el.wikipedia.org/wiki/%CE%95%CF%80%CE%B9%CF%83%CF%84%CE%AE%CE%BC%CE%B5%CF%82_%CE%BC%CE%B7%CF%87%CE%B1%CE%BD%CE%B9%CE%BA%CF%8E%CE%BD

[δ2]http://www.intertek.com/analytical-laboratories/reverse-engineering/
https://en.wikipedia.org/wiki/Deformulation

[δ3]https://www.geeksforgeeks.org/software-engineering-reverse-engineering/

[δ4]https://www.raypcb.com/pcb-reverse-engineering/

[δ5]https://transport.itu.edu.tr/docs/librariesprovider99/dersnotlari/dersnotlarimak537e/notlar/7-wireframe-and-surface-modeling.pdf?sfvrsn=4

[δ6]http://web.mit.edu/16.810/www/16.810_L4_CAE.pdf

[δ7]http://vlab.amrita.edu/?sub=3&brch=106&sim=661&cnt=1

[δ8]https://opencourses.auth.gr/modules/document/file.php

[δ9]https://3dinsider.com/structured-light-3d-scanning/

[δ10]https://www.3dnatives.com/en/3d-scanner-laser-triangulation080920174-99/

[δ11]https://www.seeedstudio.com/blog/2020/01/08/what-is-a-time-of-flight-sensor-and-how-does-a-tof-sensor-work/

[δ12]https://ems-usa.com/3d-knowledge-center/3d-scanning-knowledge-center/types-of-3d-scanners-and-3d-scanning-technologies/

[δ13]http://www.cse.uoi.gr/~palios/comp_geom/Comp_Geometry_Chapter2.pdf

[δ14]*http://oa.upm.es/32464/1/tesis-Ricardo-Picatoste.pdf*

[δ15]https://www.circuitbasics.com/basics-uart-communication/

[δ16]https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all

[δ17]https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

[δ18]http://www.eln.teilam.gr/sites/default/files/%20%CE%9A%CE%A5%CE%9A%CE%9B%CE%A9%CE%9C%CE%91%CE%A4%CE%91%20%CE%95%CE%9A%CE%A0%CE%91%CE%99%CE%94%CE%95%CE%A5%CE%A4%CE%99%CE%9A%CE%9F%20%CE%A5%CE%9B%CE%99%CE%9A%CE%9F.pdf

[δ19]https://www.electronicsforu.com/resources/difference-between-microprocessor-and-microcontroller

[δ20]https://easyeda.com/

Εικόνες

[ε1]https://www.factum-arte.com/pag/1345/photogrammetry

[ε2]https://www.revopoint3d.com/how-does-light-influence-3d-scanning/

[ε3]https://www.photonics.com/Articles/Configuring_a_3D_Triangulation_Vision_System/a62061

[ε4]https://www.seeedstudio.com/blog/2020/01/08/what-is-a-time-of-flight-sensor-and-how-does-a-tof-sensor-work/

[ε5]https://robodk.com/blog/robot-metrology-assessment/

[ε6]https://en.wikipedia.org/wiki/Convex_hull#/media/File:ConvexHull.svg

[ε7]https://www.geeksforgeeks.org/convex-hull-set-1-jarviss-algorithm-or-wrapping/

[ε8]https://en.wikipedia.org/wiki/Graham_scan#/media/File:Graham_Scan.svg

[ε9]https://reader.elsevier.com/reader/sd/pii/S0925772106000629?token=B6D82B107F29380CA73E7ED8D7156192BC29493CF40580AE1D06BEF8FDF144FC0AEE09F81BCC4B4B4DC62C5553F94E74

[ε10]https://iopscience.iop.org/article/10.1088/1742-6596/1544/1/012118/pdf

[ε11]https://en.wikipedia.org/wiki/Delaunay_triangulation#/media/File:Delaunay_circumcircles_centers.svg

[ε12]http://compbio.mit.edu/publications/C01_Amenta_Siggraph_98.pdf

[ε13]https://www.researchgate.net/figure/a-Full-step-operation-and-b-half-step-operation-for-a-two-phase-stepping-motor_fig2_3219650

[ε14]http://oa.upm.es/32464/1/tesis-Ricardo-Picatoste.pdf

[ε15]http://learnchannel-tv.com/drives/stepper-motor/torque-frequency-curve/

[ε16]https://sosteneslekule.blogspot.com/2015/11/permanent-magnet-stepper-motor.html

[ε17]https://www.electronicwings.com/sensors-modules/stepper-motor

[ε18]https://www.circuitbasics.com/basics-uart-communication/

[ε19]https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all

[ε20]https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

[ε21]https://fysikafysikh.wordpress.com/2014/10/16/%CF%83%CF%85%CE%BD%CE%B4%CE%B5%CF%83%CE%BC%CE%BF%CE%BB%CE%BF%CE%B3%CE%AF%CE%B1-%CE%B1%CE%BD%CF%84%CE%B9%CF%83%CF%84%CE%B1%CF%84%CF%8E%CE%BD/

[ε22]https://www.allaboutcircuits.com/technical-articles/understanding-i-v-curves-of-non-linear-devices/

[ε23]https://www.pololu.com/product/2133

[ε24]https://grobotronics.com/adafruit-vl6180x-time-of-flight-distance-ranging-sensor-vl6180.html?sl=en

Προγραμματισμός

[π1]https://docs.python.org/3/library/tkinter.html

[π2]https://pythonhosted.org/easygui/index.html

[π3]https://docs.python.org/3/library/subprocess.html

[π4]https://docs.python.org/3/library/multiprocessing.html

[π5]https://docs.python.org/3/library/struct.html

[π6]https://pyserial.readthedocs.io/en/latest/shortintro.html

[π7]https://docs.python.org/3/library/threading.html

[π8]https://gist.github.com/bitsgalore/901d0abe4b874b483df3ddc4168754aa

[π9]https://gist.github.com/moshekaplan/c425f861de7bbf28ef06

[π10]https://stackoverflow.com/questions/12297892/tkinter-launching-process-via-multiprocessing-creates-not-wanted-new-window

[π11]https://stackoverflow.com/questions/33970690/why-python-executable-opens-new-window-instance-when-function-by-multiprocessing

[π12] https://www.mathworks.com/matlabcentral/fileexchange/63731-surface-reconstruction-from-scattered-points-cloud-open-surfaces?focused=7941787&tab=function

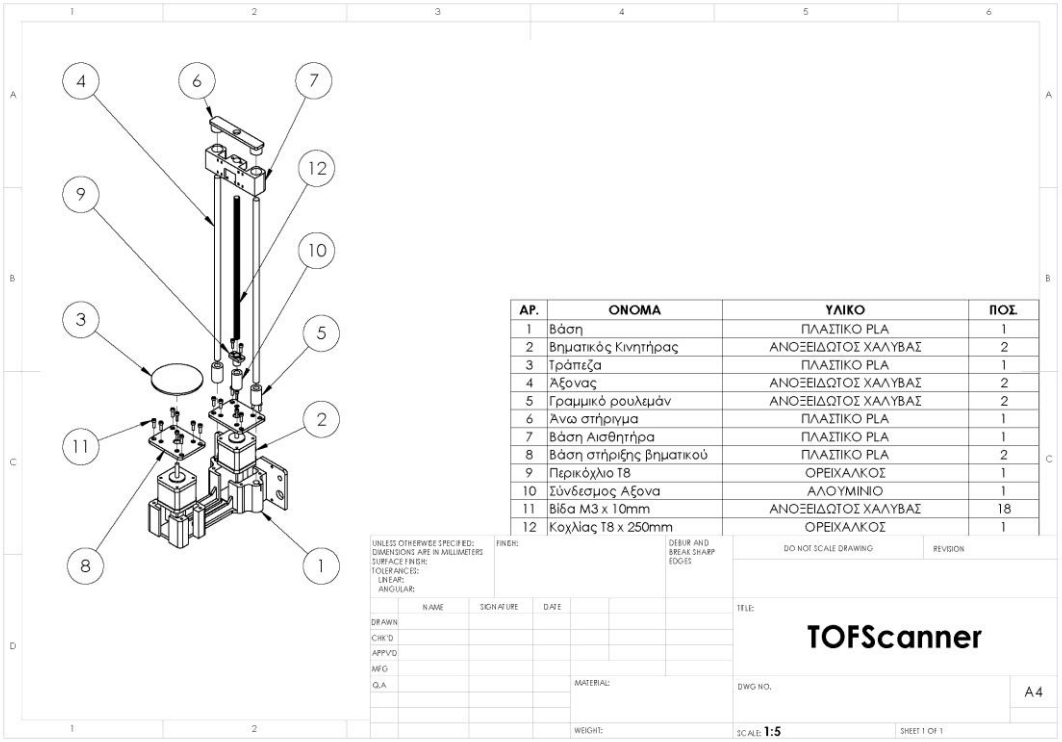
[π13]https://www.mathworks.com/matlabcentral/fileexchange/20922-stlwrite-write-ascii-or-binary-stl-files

[π14]https://stackoverflow.com/questions/45066518/nameerror-name-exit-is-not-defined

[π15]https://stackoverflow.com/questions/41884980/cx-freeze-can-not-import-numpy/51629838

ΠΑΡΑΡΤΗΜΑ Α

Μηχανολογικό σχέδιο



ΑΡ.	ΟΝΟΜΑ	ΥΛΙΚΟ	ΠΟΣ.
1	Βάση	ΠΛΑΣΤΙΚΟ PLA	1
2	Βηματικός Κινητήρας	ΑΝΟΞΕΙΔΩΤΟΣ ΧΑΛΥΒΑΣ	2
3	Τράπεζα	ΠΛΑΣΤΙΚΟ PLA	1
4	Άξονας	ΑΝΟΞΕΙΔΩΤΟΣ ΧΑΛΥΒΑΣ	2
5	Γραμμικό ρουλεμάν	ΑΝΟΞΕΙΔΩΤΟΣ ΧΑΛΥΒΑΣ	2
6	Άνω στήριγμα	ΠΛΑΣΤΙΚΟ PLA	1
7	Βάση Αισθητήρα	ΠΛΑΣΤΙΚΟ PLA	1
8	Βάση στήριξης βηματικού	ΠΛΑΣΤΙΚΟ PLA	2
9	Περικόχλιο T8	ΟΡΕΙΧΑΛΚΟΣ	1
10	Σύνδεσμος Αξονα	ΑΛΟΥΜΙΝΙΟ	1
11	Βίδα M3 x 10mm	ΑΝΟΞΕΙΔΩΤΟΣ ΧΑΛΥΒΑΣ	18
12	Κοχλίας T8 x 250mm	ΟΡΕΙΧΑΛΚΟΣ	1

UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:

FINISH:

DEBUR AND BREAK SHARP EDGES

DO NOT SCALE DRAWING

REVISION

	NAME	SIGNATURE	DATE
DRAWN			
CHK'D			
APPV'D			
MFG			
Q.A			

MATERIAL:

WEIGHT:

TITLE:

TOFScanner

DWG NO.

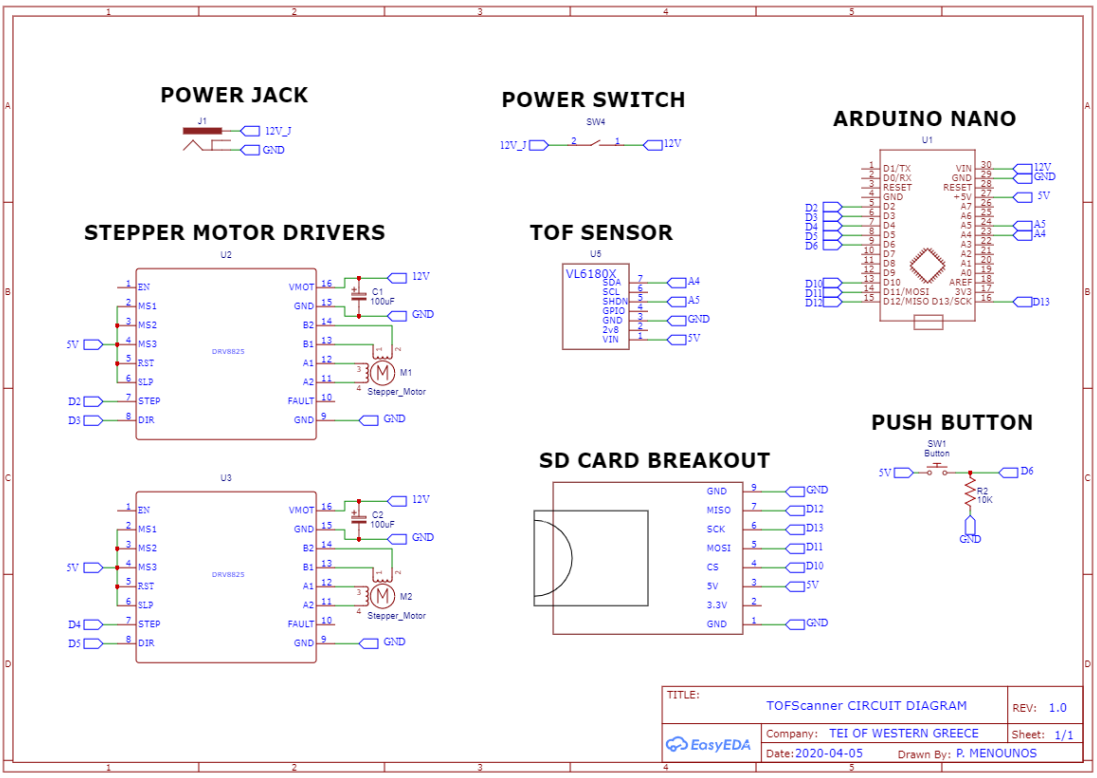
A4

SCALE: 1:5

SHEET 1 OF 1

ΠΑΡΑΡΤΗΜΑ Β

Σχηματικό διάγραμμα κυκλώματος



ΠΑΡΑΡΤΗΜΑ Γ

Πηγαίος κώδικας Python

```
from Tkinter import * [π1]
import numpy
import Tkinter as tk
import easygui [π2]
from multiprocessing import Process
import subprocess [π3]
import multiprocessing [π4]import os
import ScrolledText
import sys
from sys import exit
import math
import struct [π5]
import serial [π6]
import time
import logging
import threading [π7]

def pointcloud_process(): #υπορουτίνα υπολογισμού νέφους σημείων
    logging.info("Please wait this may take a while.....")
    sd_path_file = open(os.getenv('LOCALAPPDATA')+'\\TOFScan\\'+'sd scan file path.txt','r')
    sd_path = sd_path_file.read()
    sd_path_file.close()
    pointcloud_path_file = open(os.getenv('LOCALAPPDATA')+'\\TOFScan\\'+'pointcloud path.txt','r')
    pointcloud_path = pointcloud_path_file.read()
    pointcloud_path_file.close()
    sd_size = int(os.path.getsize(sd_path)) #μέγεθος δυαδικού αρχείου
    sd_size=sd_size-(3200) #αφαίρεση μετρήσεων τελευταίας περιστροφής
    sd_size=(sd_size/(3200))*400 #υπολογισμό αριθμού ανάγνωσης από το δυαδικό αρχείο
    pointcloud_file = open(pointcloud_path+"\\pointcloud.txt", "w")
    sd_file = open(sd_path,'rb')
    dist_from_center = 100.0 #απόσταση αισθητήρα από κέντρο περιστροφής
    angle_step = 0.9 #Γωνία ανάγνωσης απόστασης
    Z_step = 1.0 #Απόσταση ανύψωσης αισθητήρα σε mm
    Z = 0 #αρχικό ύψος αισθητήρα
    theta = 0.0 #γωνία εκκίνησης σάρωσης
    i = 0 #μετρητής βημάτων περιστροφής
    for x in range(sd_size): #ανάγνωση όλου του δυαδικού αρχείου
        g = 0 #μετρητής ανάγνωσης
        f_distance = 0 # μεταβλητή απόστασης
        total = 0 #μετρητής αθροίσματος απόστασης μέσου όρου
        while g<2: #δύο μετρήσεις
            bytes = sd_file.read(4) #ανάγνωση μέτρησης μεγέθους 4byte
            (distance,)=struct.unpack('I',bytes) #εξαγωγή ακέραιου αριθμού
            total = total + distance #πρόσθεση
            g = g + 1
        f_distance = float(total/2.0) #M.O
        if f_distance<100: #μέγιστη δεκτή απόσταση 100mm
            R = float(dist_from_center-f_distance) #υπολογισμός ακτίνας
            X=R*math.cos(math.radians(theta)) #τεταγμένη
            Y=R*math.sin(math.radians(theta)) #τετμημένη
            pointcloud_file.write(str(X) + ";" + str(Y) + ";" + str(Z) + "\n") #αποθήκευση συντεταγμένων
        theta = theta + angle_step #αύξηση γωνίας περιστροφής
        i=i+1
        if i == 400: #ολοκλήρωση περιστροφής
            theta=0 #αρχικοποίηση γωνίας
```

```
        i=0 #αρχικοποίηση μετρητή
        Z = Z + Z_step #΄ανύψωση αισθητήρα
    sd_file.close()
    pointcloud_file.close()
    logging.info("Pointcloud DONE.....")
    #root.withdraw()

def usb_scan_process(): #υπορουτίνα σάρωσης από USB
    path_file = open(os.getenv('LOCALAPPDATA')+'\\TOFScan\\'+'usb scan file path.txt','r')
    path = path_file.read()
    path_file.close()
    binary_file = open(path+"\\dataUSB.bin", "wb")
    connect_file = open(os.getenv('LOCALAPPDATA')+'\\TOFScan\\'+'connect.txt','r')
    array = connect_file.readlines() #ανάγνωση αποθηκευμένων ρυθμίσεων
    connect_file.close()
    COM_PORT = "COM"+array[0].rstrip('\n')
    BAUD_RATE = int(array[1].rstrip('\n'))
    try:
      ser=serial.Serial(COM_PORT, BAUD_RATE) #αρχικοποίηση σύνδεσης
      ser.close()
      ser.open() #άνοιγμα σειρακής θύρας
      time.sleep(2) #wait
      scan_character = 's'
      ser.write(scan_character) #αποστολή χαρακτήρα έναρξης σάρωσης
      counter = 0
      logging.info("USB Scan start.....")
      endscan = 'n'
      while str(endscan) == 'n': #επανάληψη όσο το Arduino δεν στέλνει τον χαρακτήρα τερματισμού
         bytes = ser.read(4) #ανάγνωσης 4 bytes
         (distance,)=struct.unpack("i",bytes) #εξαγωγή ακέραιου αριθμού
         binary_file.write(struct.pack("i",distance)) #αποθήκευση μέτρησης
         bytes = ser.read(4)
         (distance,)=struct.unpack("i",bytes)
         binary_file.write(struct.pack("i",distance))
         counter = counter + 1
         if counter == 400: #σε μία πλήρη περιστροφή
           endscan = ser.read(1) #έλεγχος χαρακτήρα τερματισμού
           counter = 0 #αρχικοποίηση μετρητή
      logging.info("USB Scan completed.....")
    except: #αποτυχία σύνδεσης
      logging.info("USB Scan failed.....")
      return
    #binary_file.close()
    #root.withdraw()


def save_connect_settings(): #συνάρτηση αποθήκευσης ρυθμίσεων
     open(os.getenv('LOCALAPPDATA')+'\\TOFScan\\connect.txt', 'w').close()
     saveFile = open(os.getenv('LOCALAPPDATA')+'\\TOFScan\\connect.txt','w')
     if (not str(t1.get(1.0,END)[:-1])) or (str(t1.get(1.0,END)[:-1])=='') or (str(t1.get(1.0,END)[:-1])=='\n'):
#έλεγχος συμπληρωμένων πεδίων
      saveFile.write("0\n")
     else:
      saveFile.write(str(t1.get(1.0,END)[:-1]).split()[0]+'\n') #το split διαγράφει τον χαρακτήρα '\n'
     if (not str(t2.get(1.0,END)[:-1])) or (str(t2.get(1.0,END)[:-1])=='') or (str(t2.get(1.0,END)[:-1])=='\n'):
      saveFile.write("0\n")
     else:
      saveFile.write(str(t2.get(1.0,END)[:-1]).split()[0]+'\n')
     saveFile.close()
     logging.info("New settings saved.....")
```

```
def connect(): #συνάρτηση απόπειρας σύνδεσης με το Arduino
    file = open(os.getenv('LOCALAPPDATA')+'\\TOFScan\\'+'connect.txt','r')
    array = file.readlines()
    COM_PORT = "COM"+array[0].rstrip('\n') #αφαίρεση χαρακτήρα νέας γραμμής
    BAUD_RATE = int(array[1].rstrip('\n'))
    try:
      ser=serial.Serial(COM_PORT, BAUD_RATE,timeout=5) #αρχικοποίηση σύνδεσης
      ser.close()
      ser.open() #άνοιγμα σειριακής θύρας
      time.sleep(2) #wait
      connect_character = 'c'
      ser.write(connect_character) #αποστολή χαρακτήρα σύνδεσης
      time.sleep(2)
      connect = ser.read(1) #ανάγνωση εισερχόμενου χαρακτήρα
      time.sleep(2)
      if str(connect) == 'y': #έλεγχος εισερχόμενου χαρακτήρα
        l4.config(text='Connected')
        file = open(os.getenv('LOCALAPPDATA')+'\\TOFScan\\connected.txt','w')
        file.write("0\n")
        ser.close()
        logging.info("Successfully connected.....")
      else:
        l4.config(text='Disconnected')
        open(os.getenv('LOCALAPPDATA')+'\\TOFScan\\connected.txt', 'w').close()
        ser.close()
        logging.info("Failed to connect.....")
    except: #αποτυχία σύνδεσης
      logging.info("Failed to connect.....")
      l4.config(text='Disconnected')
      open(os.getenv('LOCALAPPDATA')+'\\TOFScan\\connected.txt', 'w').close()
      ser.close()

def usb_scan(): #συνάρτηση σάρωσης από USB
   if os.path.getsize(os.getenv('LOCALAPPDATA')+'\\TOFScan\\'+"connected.txt") > 0: #έλεγχος
σύνδεσης με την συσκευή
       filepath = easygui.diropenbox() #επιλογή διαδρομής αποθήκευσης μετρήσεων
       if filepath != None:
           savepathfile = open(os.getenv('LOCALAPPDATA')+'\\TOFScan\\'+'usb scan file path.txt','w')
           savepathfile.write(filepath)
           savepathfile.close()
           logging.info("USB Scan request.....")
           t1 = threading.Thread(target=usb_scan_process, args=[])
           t1.start() #έναρξη υπορουτίνας σάρωσης από USB
           #p = Process(target=usb_scan_process,args=[])
           #p.start()
   else: #προειδοποιητικό μήνυμα σύνδεσης πρώτα
       logging.info("Try to connect first.....")

def import_binary(): #συνάρτηση εισαγωγής αρχείου μετρήσεων
   filepath = easygui.fileopenbox() #open window to select file and save path
   if filepath != None:
       savepathfile = open(os.getenv('LOCALAPPDATA')+'\\TOFScan\\'+'sd scan file path.txt','w')
       savepathfile.write(filepath)
       savepathfile.close()
       logging.info("Scan file imported.....")

def import_pointcloud(): #συνάρτηση εισαγωγής αρχείου νέφους σημείων
   filepath = easygui.fileopenbox() #open window to select file and save path
   if filepath != None:
```

```
        savepathfile = open(os.getenv('LOCALAPPDATA')+'\\TOFScan\\'+'pointcloudpath.txt','w')
        savepathfile.write(filepath)
        savepathfile.close()
        logging.info("Pointcloud file imported.....")

def stl(): #συνάρτηση κλήσης κώδικα MATLAB για την ανοικοδόμηση της επιφάνειας
    filepath = easygui.diropenbox() #open window to select file and save path
    if filepath != None:
        savepathfile = open(os.getenv('LOCALAPPDATA')+'\\TOFScan\\'+'stlpath.txt','w')
        savepathfile.write(filepath)
        savepathfile.close()
        logging.info("Surface reconstruction request.....")
        logging.info("Please wait this may take a while.....")
        p2 = subprocess.Popen("surface_reconstruction.exe")

def pointcloud(): #συνάρτηση νέφους σημείων
    filepath = easygui.diropenbox() #open window to select file and save path
    if filepath != None:
        savepathfile = open(os.getenv('LOCALAPPDATA')+'\\TOFScan\\'+'pointcloud path.txt','w')
        savepathfile.write(filepath)
        savepathfile.close()
        logging.info("Pointcloud calculation request.....")
        p = threading.Thread(target=pointcloud_process,args=[]) #κλήση υπορουτίνας υπολογισμού
νέφους σημείων
        p.start()


class TextHandler(logging.Handler): # [π8]
    #This class allows you to log to a Tkinter Text or ScrolledText widget
    #Adapted from Moshe Kaplan: [π4]    def __init__(self, text):
        # run the regular Handler __init__
        logging.Handler.__init__(self)
        # Store a reference to the Text it will log to
        self.text = text

    def emit(self, record):
        msg = self.format(record)
        def append():
            self.text.configure(state='normal')
            self.text.insert(tk.END, msg + '\n')
            self.text.configure(state='disabled')
            # Autoscroll to the bottom
            self.text.yview(tk.END)
        # This is necessary because we can't modify the Text from other threads
        self.text.after(0, append)

root = Tk() #δημιουργία βασικού παραθύρου
root.title("TOFScanner")#τίτλος παραθύρου
root.geometry('{}x{}'.format(510, 320))#μέγεθος παραθύρου
root.resizable(width=False, height=False)

st = ScrolledText.ScrolledText(root, bd=1.8,width=60,height=7, state='disabled')
st.configure(font='TkFixedFont')
st.place(x=5,y=190)

connect_frame = Frame(root,bd=2, relief=GROOVE)
connect_frame.place(x=20,y=20)
l1=Label(connect_frame,text="PORT",width=11,height=1,font=("Calibri", 12),relief=FLAT) #δημιουργία
label
l1.grid(row=0, column=0,padx=(0, 0),pady=(0, 0))
```

```
l2=Label(connect_frame,text="BAUDRATE",width=11,height=1,font=("Calibri", 12),relief=FLAT)
l2.grid(row=1, column=0,padx=(0, 0), pady=(0, 0))
l3=Label(connect_frame,text="STATUS:",width=11,height=1,font=("Calibri", 12),relief=FLAT)
l3.grid(row=2, column=0, padx =(0, 0), pady=(0, 0))
l4=Label(connect_frame,text="Disconnected",width=11,height=1,font=("Calibri", 12),relief=FLAT)
l4.grid(row=2, column=1,padx=(0, 5), pady=(5, 5))
t1=Text(connect_frame, wrap=NONE, bd=1.8,width = 7, height = 1,font=("Calibri", 12)) #δημιουργία
κελιών
t1.grid(row=0, column=1, padx=(0,5), pady=(8,5))
t2=Text(connect_frame, wrap=NONE, bd=1.8,width = 7, height = 1,font=("Calibri", 12))
t2.grid(row=1, column=1, padx=(0,5), pady=(5,5))
b1=Button(connect_frame, text="SAVE",font=("Calibri", 12),width = 9, height = 1,command =
save_connect_settings) #δημιουργία κουμπιών
b1.grid(row=3, column=0, padx=(0, 0), pady=(5, 8))
b2=Button(connect_frame, text="CONNECT",font=("Calibri", 12),width = 9, height = 1,command =
connect)
b2.grid(row=3, column=1, padx=(0, 0), pady=(5, 8))


buttons_frame = Frame(root,bd=2, relief=GROOVE)
buttons_frame.place(x=250,y=20)
b3=Button(buttons_frame, text="USB_SCAN",font=("Calibri", 12),width = 12, height = 1,command =
usb_scan)
b3.grid(row=0, column=0, padx=(8, 8), pady=(14, 0))
b4=Button(buttons_frame, text="BINARY",font=("Calibri", 12),width = 12, height = 1,command =
import_binary)
b4.grid(row=0, column=1, padx=(0, 8), pady=(14, 0))
b5=Button(buttons_frame, text="POINTCLOUD",font=("Calibri", 12),width = 12, height = 1,command =
pointcloud)
b5.grid(row=1, column=0, padx=(8, 8), pady=(14, 0))
b6=Button(buttons_frame, text="IMPORT_PCD",font=("Calibri", 12),width = 12, height = 1,command =
import_pointcloud)
b6.grid(row=1, column=1, padx=(0, 8), pady=(14, 0))
b7=Button(buttons_frame, text="EXPORT_STL",font=("Calibri", 12),width = 12, height = 1,command =
stl)
b7.grid(row=3, columnspan=2, padx=(8, 8), pady=(14, 13))

# Create textLogger
text_handler = TextHandler(st)

# Logging configuration
logging.basicConfig(filename=os.getenv('LOCALAPPDATA')+'\\TOFScan'+'\\log.log',
level=logging.INFO,
format='%(asctime)s - %(levelname)s - %(message)s')

#Add the handler to logger
logger = logging.getLogger()
logger.addHandler(text_handler)

with open(os.getenv('LOCALAPPDATA')+'\\TOFScan\\connect.txt') as file:
  array = file.readlines() #ανάγνωση ρυθμίσεων και εγγραφή στα κελιά στο ξεκίνημα
  if array[0]!=' \n': t1.insert(END, array[0]) #if fixes a bug
  if array[1]!=' \n': t2.insert(END, array[1])
  file.close()
  logging.info("TOFScanner app.....")

class AppUI(Frame): #δημιουργία μενού

   def __init__(self, master=None):
      Frame.__init__(self, master, relief=SUNKEN, bd=2)
```

```
        self.menubar = Menu(self)

        menu = Menu(self.menubar, tearoff=0)
        self.menubar.add_cascade(label="Exit", command=exit) #τερματισμό εφαρμογής

        try:
            self.master.config(menu=self.menubar)
        except AttributeError:
            # master is a toplevel window (Python 1.4/Tkinter 1.63)
            self.master.tk.call(master, "config", "-menu", self.menubar)

def main():
  app = AppUI(root)
  app.pack()
  root.mainloop()

if __name__ == "__main__": # [π10]
  multiprocessing.freeze_support() #πολύ σημαντικό για να λειτουργήσει η μεταγλωττισμένη έκδοση
[π11]
main()
```

Πηγαίος κώδικας MATLAB

```
clc
```

```
clear
localppdata = getenv('LOCALAPPDATA'); %LOCALAPPDATA
localppdata = strrep(localppdata,'\','/');
foldername = '/TOFScan';
stlfilename = '/stlscan.stl';
stlpathfilename = '/stlpath.txt';

path2 = strcat(localppdata,foldername,stlpathfilename);
fid1 = fopen(path2);
stlpath = fgetl(fid1);
stlpath = strrep(stlpath,'\','/');
fclose(fid1);
stlpath = strcat(stlpath,stlfilename)
pointcloudpath = '/pointcloudpath.txt'
path1 = strcat(localppdata,foldername,pointcloudpath);
fid = fopen(path1);
pointcloudpath = fgetl(fid);
pointcloudpath = strrep(pointcloudpath,'\','/');
fclose(fid);
XYZ = dlmread(pointcloudpath)
ptCloud = pointCloud(XYZ);
ptCloud = removeInvalidPoints(ptCloud);
ptCloud = pcdenoise(ptCloud);
X = ptCloud.Location(:,1);
Y = ptCloud.Location(:,2);
Z = ptCloud.Location(:,3);
V = [X Y Z];
V = unique(V,'rows');
[F]=MyCrustOpen(V); [π12]
stlwrite(stlpath,F,V) [π13]
```

Πηγαίος κώδικας Arduino

```
#include <SPI.h>
#include "SdFat.h"
#include <Wire.h>
#include "Adafruit_VL6180X.h"

Adafruit_VL6180X vl = Adafruit_VL6180X();
SdFat SD;
#define SD_CS_PIN SS //αντιστοιχεί στο Pin 10
File myFile;

void scan(int mode); //υπορουτίνα σάρωσης
int SD_setup(); //υπορουτίνα αρχικοποίησης κάρτας μνήμης
char income,connect_message = 'y', check_message = 'n';

int32_t distance; //μεταβλητή μετρούμενης απόστασης
unsigned long time_now = 0; //μεταβλητή χρόνου

void setup() {
  Serial.begin(9600);
  pinMode(6,INPUT);
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(SS, OUTPUT);
  vl.begin();
}

void loop() {
  if(digitalRead(6)){ //Έλεγχος εάν πατηθεί το κουμπί έναρξης σάρωσης σε κάρτα μνήμης
     if(SD_setup()){ //Εάν είναι επιτυχής η αρχικοποίηση της κάρτας μνήμης
        scan(0); //Τότε έναρξη σάρωσης σε κάρτα μνήμης, mode=0
        myFile.close(); //Κλείσιμο αρχείου μετά το πέρας της σάρωσης
     }
   }
}

void scan(int mode){
  long
steps=0,micro_steps=0,z_steps=0,total_zsteps=0,f=0,steps_per_rev=400,period=1,read_times=0,micr
oseconds=500;
  uint8_t sensor;
  boolean first=true,terminate=false;
  while(terminate == false){ //Όσο δεν ικανοποιείται η συνθήκη τερματισμού σάρωσης κάνε επανάληψη
     terminate=true; //Θέτεται ο τερματισμός σάρωσης αληθής
     while(steps<steps_per_rev){ //Κάνε επανάληψη για μία περιστροφή
        time_now = millis(); //Παρών χρόνος
        read_times=0; //Μετρητής ανάγνωσης αισθητήρα
        while(read_times<2){ //Πάρε δύο μετρήσεις από τον αισθητήρα
           sensor = vl.readRange();
           distance = sensor;
           if (distance <100) { //Εάν η μέτρηση είναι τουλάχιστον μία φορά μικρότερη από 100mm σε μία
περιστροφή τότε συνθήκη τερματισμού ψευδής
              terminate=false;
           }
           if(mode == 1){ //Σάρωση από USB
              Serial.write((const uint8_t*)&distance,sizeof(distance));
           }else{ //Σάρωση σε κάρτα μνήμης
              myFile.write((const uint8_t *)&distance, sizeof(distance));
           }
```

```
        read_times++;
      }
      while(millis() < time_now + period){ //Αναμονή χρόνου period
        ;//wait
      }
      while(micro_steps<16){ //Εκτέλεση περιστροφής κατα 0.9 μοίρες
        if(first=false){
          delayMicroseconds(microseconds);
        }
        first=false;
        digitalWrite(2,HIGH);
        delayMicroseconds(microseconds);
        digitalWrite(2,LOW);
        micro_steps++;
      }
      first=true;
      micro_steps=0;
      steps++;
    }
    steps=0;
    micro_steps=0;
    if(terminate == false){ //Εφόσον δεν έχει ολοκληρωθεί η σάρωση
      while(z_steps<800){ //ανύψωση αισθητήρα κατα 1mm
        delayMicroseconds(100);
        digitalWrite(4,HIGH);
        delayMicroseconds(100);
        digitalWrite(4,LOW);
        z_steps++;
        total_zsteps++;
      }
    }else{ //Εφόσον έχει ολοκληρωθεί η σάρωση
      digitalWrite(5,HIGH);
      while(z_steps<total_zsteps){ //Μετακίνηση αισθητήρα στην αρχική θέση
        delayMicroseconds(100);
        digitalWrite(4,HIGH);
        delayMicroseconds(100);
        digitalWrite(4,LOW);
        z_steps++;
      }
      digitalWrite(5,LOW);
      total_zsteps=0;
    }
    if(mode == 1){ //Εφόσον γίνεται σάρωση από USB
                        if(terminate==false){ //Εάν δεν έχει τερματιστεί ακόμα
                                check_message = 'n'; //Στέλνεται μήνυμα ανύψωσης αισθητήρα
                        }else{ //Εάν τερματίστηκε η διαδικασία
                                check_message = 'y'; //Στέλνεται μήνυμα ολοκλήρωσης
                        }
                        Serial.write((const uint8_t*)&check_message,sizeof(check_message));
            }
    delayMicroseconds(600);
    z_steps=0;
    micro_steps=0;
  }
}

int SD_setup()
{
  int success;
  if(!SD.begin(SD_CS_PIN)){ //Έλεγχος σύνδεδεμένης κάρτας μνήμης
```

```
    success=0;
  }else{
    delay(1200);
    myFile = SD.open("dataSD.bin", FILE_WRITE); //Δημιουργία αρχείου στην κάρτα μνήμη
    success=1;
  }
  return success;
}

void serialEvent(){ //Ρουτίνα ανίχνευσης εισερχόμενων δεδομένων από USB
    income=Serial.read(); //Ανάγνωση εισερχόμενου byte
    if(income == 'c'){ //Εάν είναι c, τότε στέλνεται πίσω ο χαρακτήρας σύνδεσης
      Serial.write((const uint8_t*)&connect_message,sizeof(connect_message));
    }
    if (income=='s'){ //Αλλιώς ξεκινάει η διαδικασία σάρωσης από USB, mode=1
       scan(1);
    }
}
```

<u>Κώδικας Python μεταγλώττισης cxFreeze</u>

```
import sys
from sys import exit [π14]
import numpy.core._methods
import numpy.lib.format
from cx_Freeze
import setup, Executable

build_exe_options = {"packages": ["os", "numpy"], "includes": ["numpy"]} [π15]
base = None
if (sys.platform == "win32"):
    base = "Win32GUI"

exe = Executable(
        script = "TOFScan.py",
        targetName = "TOFScan.exe",
        base = base
        )

setup(
    name = "TOFScan",
    version = "1.0.0",
    description = "3D Scanner firmware",
    author = "Panagiotis Menounos",
    options={"build_exe": build_exe_options},
    executables = [exe]
)
```

<u>Κώδικας συμπίεσης Pascal του Inno Setup</u>

```
; Script generated by the Inno Setup Script Wizard.
; SEE THE DOCUMENTATION FOR DETAILS ON CREATING INNO SETUP SCRIPT FILES!

[Setup]
; NOTE: The value of AppId uniquely identifies this application.
; Do not use the same AppId value in installers for other applications.
; (To generate a new GUID, click Tools | Generate GUID inside the IDE.)
AppId={{AFEF1D22-6433-42E0-A573-7BD372A95287}
AppName=TOFScan
AppVersion=1.0
;AppVerName=setup 1.0
AppPublisher=TEI
DefaultDirName={pf}\TOFScan
DisableProgramGroupPage=yes
OutputDir=C:\Users\user\Desktop\PYTHON\TOF
OutputBaseFilename=TOFScan
Compression=lzma
SolidCompression=yes

[Languages]
Name: "english"; MessagesFile: "compiler:Default.isl"
Name: "greek"; MessagesFile: "compiler:Languages\Greek.isl"

[Tasks]
Name: "desktopicon"; Description: "{cm:CreateDesktopIcon}"; GroupDescription:
"{cm:AdditionalIcons}"; Flags: unchecked
Name: "quicklaunchicon"; Description: "{cm:CreateQuickLaunchIcon}"; GroupDescription:
"{cm:AdditionalIcons}"; Flags: unchecked; OnlyBelowVersion: 0,6.1

[Files]
Source: "C:\Users\user\Desktop\PYTHON\TOF\Files to compress\Compiled\TOFScan.exe"; DestDir:
"{app}"; Flags: ignoreversion
Source: "C:\Users\user\Desktop\PYTHON\TOF\Files to compress\Compiled\*"; DestDir: "{app}"; Flags:
ignoreversion recursesubdirs createallsubdirs
Source: "C:\Users\user\Desktop\PYTHON\TOF\Files to compress\TOFScan\*"; DestDir:
"{localappdata}\TOFScan"; Flags: ignoreversion recursesubdirs createallsubdirs
; NOTE: Don't use "Flags: ignoreversion" on any shared system files

[Icons]
Name: "{commonprograms}\TOFScan"; Filename: "{app}\TOFScan.exe"
Name: "{commondesktop}\TOFScan"; Filename: "{app}\TOFScan.exe"; Tasks: desktopicon
Name: "{userappdata}\Microsoft\Internet Explorer\Quick Launch\setup"; Filename:
"{app}\TOFScan.exe"; Tasks: quicklaunchicon

[Run]
Filename: "{app}\TOFScan.exe"; Description: "{cm:LaunchProgram,setup}"; Flags: nowait postinstall
skipifsilent
```