# Αυτοματοποίηση της προετοιμασίας παραγωγής (setup) εργαλειομηχανής συγκόλλησης ηλιακών συλλεκτών με laser

Σωτήρχος Γεώργιος

Επιβλέπων: Βοσνιάκος Γεώργιος-Χριστόφορος

Σχολή Μηχανολόγων Μηχανικών,
Εθνικό Μετσόβιο Πολυτεχνείο

14 Οκτωβρίου 2019

## Περίληψη

Σε μία μονάδα παραγωγής στην οποία κατασκευάζονται σημαντικές ποσότητες διαφόρων παραλλαγών ηλιακών συλλεκτών από πολλαπλές παραγγελίες κρίνεται σημαντικό να ελαχιστοποιηθούν οι χρόνοι προετοιμασίας διατηρώντας συγχρόνως την υψηλή ποιότητα του προϊόντος. Σε αυτήν την διπλωματική εργασία περιγράφεται ο σχεδιασμός μίας αυτόματα ρυθμιζόμενης διάταξης για την συγκράτηση του προϊόντος σε σταθμό συγκόλλησης με laser. Ο τελικός στόχος είναι η πλήρης αντικατάσταση την υπάρχουσας χειρωνακτικής μεθόδου προετοιμασίας με την σχεδιασμένη διάταξη χωρίς να θυσιαστεί η ποιότητα ή η ποικιλία του προϊόντος. Κατά το στάδιο του προσδιορισμού του προβλήματος καταγράφηκαν οι ανάγκες με βάση τις οποίες προσδιορίστηκαν οι τεχνικές προδιαγραφές που πρέπει να πληροί ο σχεδιασμός, οι οποίες περιλαμβάνουν γεωμετρικούς περιορισμούς και απαιτούμενο εύρος κίνησης. Παράχθηκαν κάποιες σχεδιαστικές λύσεις σε πρώιμο στάδιο και εξετάσθηκαν ως προς την ικανοποίηση των προδιαγραφών ώστε να προκριθεί τελικά η προτιμότερη για λεπτομερή σχεδιασμό. Η επιλεγμένος σχεδιασμός αναπτύχθηκε περαιτέρω και βελτιώθηκε με χρήση λογισμικού CAD. Επιπρόσθετα, αναπτύχθηκε το λογισμικό της διεπιφάνειας μέσω της οποίας ένας χειριστής μπορεί να προετοιμάσει τη διάταξη εισάγοντας μόνο τα απαραίτητα γεωμετρικά χαρακτηριστικά ενός ηλιακού συλλέκτη. Για επαλήθευση της λειτουργικότητας και της ευχρηστίας της διάταξης επιλέχθηκαν, κατασκευάστηκαν, και εγκαταστάθηκαν τα απαραίτητα μέλη της πάνω σε δοκιμαστική τράπεζα. Μέσω των δοκιμών διαπιστώθηκε ότι η διάταξη μπορεί να χρησιμοποιηθεί με ασφάλεια και να ρυθμιστεί επιτυχώς με την απαραίτητη ακρίβεια. Η σχεδιασμένη διάταξη περιγράφεται λεπτομερώς στην παρούσα εργασία.



## Abstract

In a production facility manufacturing significant quantities of multiple variations of solar absorbers from multiple orders it is considered important to minimize the setup times while preserving high product quality. In this diploma thesis an automated configurable fixture for a laser welding machine is designed as a solution to this problem. The final purpose is to completely replace the current manual setup method with the designed fixture without sacrificing quality or flexibility in product variation. During problem definition are recorded the needs through which the technical standards the design has to satisfy are determined, which involve geometric limitations and required range of motion. Some initial rough concept models are generated and checked against the specifications in order to qualify the most promising for detail design. The selected design is developed and optimized with the use of CAD software. Additionally a software interface is developed through which the operator is able to configure the physical fixture by inputting only the necessary geometric properties of a solar absorber. Components necessary for determining the successful use and installation of the final design are selected, manufactured, installed and tested. The tests involved determining whether the fixture could be safely used, and configured successfully with the necessary precision. This paper presents the developed fixture in more detail.

# Περιεχόμενα

# 1 Εισαγωγή

Η διαδεδομένη χρήση εργαλειομηχανών και κέντρων κατεργασιών με αριθμητικό έλεγχο, σε συνδυασμό με την εισαγωγή των τεχνολογιών πληροφορικής στους χώρους παραγωγής, έχουν οδηγήσει σε μία στροφή από το πρότυπο της μαζικής παραγωγής σε αυτό της μαζικής εξατομίκευσης (mass customization) Τα σημερινά συστήματα παραγωγής καλούνται να παράξουν μεγάλο αριθμό από μικρές παρτίδες διαφορετικών προϊόντων ή παραλλαγών με οικονομικά αποδοτικές μεθόδους. Συγχρόνως η ποιότητα του προϊόντος πρέπει να παραμένει υψηλή ώστε να παραμένει ανταγωνιστικό.

Για να είναι σε θέση οι σημερινές βιομηχανίες να κατασκευάσουν ανταγωνιστικά τα τελικά προϊόντα και να διατηρήσουν κυρίαρχη θέση στην παγκόσμια αγορά χρειάζεται να επενδύσουν στην ευελιξία και στην υψηλή ποιότητα. Η ευελιξία εκφράζει πόσο ευαίσθητο σε μεταβολές μπορεί να είναι ένα περιβάλλον παραγωγής. Παράλληλα η βελτίωση της ποιότητας των προϊόντων απαιτεί τη χρήση εξειδικευμένου εξοπλισμού που εξασφαλίζει ελάχιστη απόκλιση από τα επιθυμητά χαρακτηριστικά. Ως αποτέλεσμα των παραπάνω σημαντικό μέρος της έρευνας και της ανάπτυξης στον τομέα της παραγωγής εστιάζει στην δημιουργία τεχνολογιών και εξοπλισμού που διαρκώς ωθεί τα όρια της ευελιξίας και των χρόνων κύκλου, ελαχιστοποιώντας το κόστος παραγωγής, χωρίς να θυσιάζεται η ποιότητα του τελικού προϊόντος. Οι συγκρατήσεις (fixtures) αποτελούν αναπόσπαστο μέρος των συστημάτων παραγωγής και συνεπώς διατηρούν σημαντικό ερευνητικό ενδιαφέρον.

Αυτή η εργασία περιλαμβάνει την μελέτη και τον σχεδιασμό μίας διάταξης συγκράτησης με δυνατότητα αυτόματης ρύθμισης. Στην ενότητα 2 βρίσκεται η βιβλιογραφική ανασκόπηση των υπαρχόντων μεθόδων και προτάσεων σε αντίστοιχα προβλήματα. Στην ενότητα 3 παρατίθεται το πρόβλημα, τα ποσοτικά χαρακτηριστικά του, η υπάρχουσα μέθοδος προς βελτίωση, και ο προσδιορισμός των τεχνικών χαρακτηριστικών με βάση τις οποίες θα γίνει ο σχεδιασμός. Στην ενότητα 4 περιγράφεται η διαδικασία σχεδιασμού, από την αρχική παραγωγή πρώιμων ιδεών, την επιλογή της κατάλληλης, τον λεπτομερή σχεδιασμό για την κάλυψη των απαιτήσεων, έως και την περιγραφή της χρήσης της διεπιφάνειας ρύθμισης από τους χειριστές. Στην ενότητα 5 περιγράφεται ο τρόπος αξιολόγησης της διάταξης, η κατασκευή και δοκιμή της, τα ποσοτικά συμπεράσματα από τη χρήση της καθώς και κάποιες προτάσεις περαιτέρω βελτίωσης για μελλοντική χρήση.

# 2 Βιβλιογραφική ανασκόπηση

Ως ευέλικτο σύστημα παραγωγής ορίζεται μία παραγωγική μονάδα ικανή να παράξει ένα εύρος διακριτών προϊόντων με ελάχιστη χειρωνακτική παρέμβαση [1]. Αποτελείται από σταθμούς παραγωγής διασυνδεδεμένους με σύστημα μεταφοράς και διαχείρισης των υλικών, και λειτουργεί, ιδανικά, υπό πλήρη προγραμματιζόμενο έλεγχο. Για να είναι τέτοιο σύστημα πραγματικά ευέλικτο θα πρέπει και όλα τα υποσυστήματά του να είναι ευέλικτα. Ένα μείζον εμπόδιο σε αυτήν την προσπάθεια είναι η αδυναμία των συστημάτων συγκράτησης (fixtures) να είναι ευέλικτα [2].

Με συστήματα «ευέλικτης» συγκράτησης [3] (flexible fixturing) είναι δυνατό να συγκρατηθούν τεμάχια διαφόρων σχημάτων και μεγεθών υποκείμενα σε ποικιλία εξωτερικών φορτίσεων αναλόγως την εκάστοτε μέθοδο παραγωγής. Τα πλεονεκτήματα χρήσης τέτοιων συστημάτων είναι κατ΄ αρχάς η μείωση χρόνου και κόστους για τον σχεδιασμό εξειδικευμένων συγκρατήσεων και κατ΄ επέκταση η μείωση του έμμεσου κόστους που συνδέεται με την αποθήκευση και ανάκτηση των πολλαπλών συγκρατήσεων οι οποίες χρειάζεται να τοποθετούνται στο συντομότερο δυνατό χρόνο.

Οι συγκρατήσεις αποτελούνται συχνότερα από τρία μέλη: τους «εντοπιστές» (locators), τις αρπάγες (clamps), και τις βάσεις τους (baseplate) [4]. Οι «εντοπιστές» τοποθετούν με ακρίβεια και επαναληψιμότητα το τεμάχιο, και οι αρπάγες το συγκρατούν ενάντια στο φορτίο κατεργασίας και περιορίζουν τις ταλαντώσεις, ενώ η βάση αποτελεί την πλατφόρμα πάνω στην οποία αυτά στερεώνονται [5]. Οι ευέλικτες συγκρατήσεις διακρίνονται σε τρεις κατηγορίες: τις «σπονδυλωτές» (modular), τις επαναρυθμιζόμενες (reconfigurable), και τις σύμμορφες (comformable) [6].

Τα συστήματα modular συγκρατήσεων συνήθως αποτελούνται από μία βάση με εσοχές, κυλινδρικές ή T-slot πάνω στις οποίες τοποθετούνται τα κατάλληλα στοιχεία κατά περίπτωση [7]. Οι οπές είναι διανοιγμένες με υψηλή ακρίβεια και συχνά τα στοιχεία είναι ογκώδη για να εξασφαλίζουν στιβαρότητα. Ο σχεδιασμός και η χειρωνακτική προετοιμασία ενός τέτοιου συστήματος μπορεί να διαρκέσει ώρες ακόμα και από έμπειρους χειριστές. Δεν έχει πραγματοποιηθεί σημαντική έρευνα προς την κατεύθυνση αυτοματοποιημένων modular συγκρατήσεων πέρα από λίγες προσεγγίσεις μεθοδολογιών [8], [9] και αυτόματου σχεδιασμού τους [10].

Οι επαναρυθμιζόμενες συγκρατήσεις μπορούν να ρυθμίζονται για να εξυπηρετούν διάφορα τεμάχια. Προσφέρουν ταχείς χρόνους προετοιμασίας και χρησιμοποιούνται ευρέως σε επαναρυθμίσιμα συστήματα παραγωγής (Reconfigurable Manufacturing Systems). Το βασικό μειονέκτημά τους είναι ο περιορισμένος αριθμός διαφορετικών τεμαχίων που είναι σε θέση να συγκρατήσουν. Συχνά περιορίζονται σε παραλλαγές εντός μίας οικογένειας τεμαχίων, μία μορφή ευελιξίας χαρακτηριζόμενη ως «εξατομικευμένη» (customized flexibility). Παρόλο που δεν είναι τα καταλληλότερα για τις ανάγκες της μαζικής εξατομίκευσης, στην εφαρμογή που μελετάται σε αυτή την εργασία αποτελούν μία πιθανή κατηγορία λύσεων.

Οι σύμμορφες συγκρατήσεις έχουν την ικανότητα να λάβουν τη γεωμετρία των τεμαχίων που συγκρατούν. Χαρακτηριστικά παραδείγματα είναι είναι αυτά της «μήτρας καρφιών» (pin matrix) και των συγκρατητών αλλαγής φάσης [6]. Η αρχή λειτουργίας τους είναι η βύθιση των μη κατεργάσιμων μερών ενός τεμαχίου, και η στερεοποίηση ενός ειδικού υγρού γύρω από αυτό. Με αυτόν τον τρόπο μπορούν να συγκρατηθούν τεμάχια εύθραυστα ή ιδιαίτερης γεωμετρίας. Βασικό μειονέκτημα είναι η δυσκολία εξόλκευσης του τεμαχίου η οποία συχνά απαιτεί χρόνο και ενέργεια, ενώ κοπτικά και ψυκτικά υγρά ενδέχεται να διαλυθούν μέσα στο

ειδικό υγρό μειώνοντας σταδιακά την ποιότητά του [11].

Η διαδικασία σχεδιασμού μίας διάταξης συγκράτησης ενός δεδομένου συνόλου προϊόντων ή μίας κατεργασίας πραγματοποιείται σε τέσσερις φάσεις. Οι φάσεις είναι οι εξής: σχεδιασμός της «προετοιμασίας» (setup planning), σχεδιασμός συγκράτησης (fixture planning), σχεδίαση της μονάδας (unit design), και επαλήθευση (verification) [12]. Ο σχεδιασμός της προετοιμασίας αφορά την ομαδοποίηση των προς κατασκευή χαρακτηριστικών του τεμαχίου για το οποία δεν χρειάζεται επανατοποθέτηση ώστε να προσδιοριστούν τα απαραίτητα setups. Ο σχεδιασμός των συγκρατήσεων περιλαμβάνει την επιλογή της κατάλληλης μεθόδου συγκράτησης για κάθε setup ώστε να εξασφαλίζονται οι εξής απαιτήσεις: περιορισμός κίνησης, ανοχές, αποφυγή πρόσκρουσης, χρηστικότητα, και οικονομία. Αυτά προσδιορίζονται μέσω των διαδικασιών ανάλυσης και σύνθεσης. Κατά τη σύνθεση μοντελοποιείται η συγκράτηση και προσδιορίζεται το σύνολο των μεταβλητών, ενώ κατά την σύνθεση επιλέγονται οι κατάλληλες τιμές των μεταβλητών με τη βοήθεια των μοντέλων της ανάλυσης [13].

Η φάση σχεδίασης της μονάδας περιλαμβάνει τον σχεδιασμό των «εντοπιστών» και των αρπαγών με βάση τα αποτελέσματα από τη φάση σχεδιασμού συγκράτησης. Αυτό συμπεριλαμβάνει τον σχεδιασμό concepts και τον λεπτομερή σχεδιασμό. Η φάση της επαλήθευσης αφορά την επαλήθευση ικανοποίησης των απαιτήσεων από την τελική σχεδίαση. Αυτό μπορεί να πραγματοποιηθεί είτε μέσω προσομοίωσης είτε με φυσική δοκιμή [12].

# 3 Προσδιορισμός προβλήματος

## 3.1 Καταγραφή αναγκών

Στην εταιρία Prime Laser Technology A.B.E.E. παράγονται ηλιακοί συλλέκτες με ποικιλία διαστάσεων. Οι πρώτες ύλες που χρησιμοποιούνται είναι χαλκοσωλήνες διαμέτρων Φ8, Φ15, και Φ22 και φύλλα τα οποία μπορεί να είναι χάλκινα, TSS, αλουμινένια βαμμένα, απορροφητικά πάχους 0.5 mm ή 0.4 mm. Με το σύστημα παραγωγής που χρησιμοποιείται ικανοποιούνται σε εβδομαδιαία βάση πολλαπλές παραγγελίες διαφόρων παραλλαγών συλλεκτών. Παράγεται το κατάλληλο πλέγμα σωληνών με κοπή και συγκόλληση τους, κόβονται τα φύλλα σε κατάλληλο μέγεθος, και συγκολλούνται μεταξύ τους σε μηχανή συγκόλλησης με laser.

Οι τύποι προϊόντων που κατασκευάζονται είναι τέσσερις. Ο συνηθέστερος τύπος είναι ο «κατακόρυφος» συλλέκτης στον οποίο σε μακρόστενο φύλλο αλουμινίου συγκολλάται πλέγμα σωληνών με τους riser κατά μήκος της μεγάλης διάστασης του φύλλου και οι header κάθετα, κατά μήκος της μικρής διάστασης. Παρόμοιος είναι ο τύπος του «οριζόντιου» συλλέκτη ο οποίος διαφέρει από τον προηγούμενο στο ότι οι riser είναι παράλληλα στη μικρή διάσταση του φύλλου ενώ οι header στη μεγάλη. Σπανιότερα συναντώνται και οι περιπτώσεις του συλλέκτη «με φινάκια» και του συλλέκτη με «μαίανδρο». Ο πρώτος είναι μορφολογικά παρόμοιος με τον «κατακόρυφο» με τη διαφορά ότι αντί για ένα φύλλο χρησιμοποιείται σειρά από λωρίδες («φινάκια») ενώ στο δεύτερο οι riser του πλέγματος έχουν (μία τυποποιημένη) γεωμετρία μαιάνδρου.

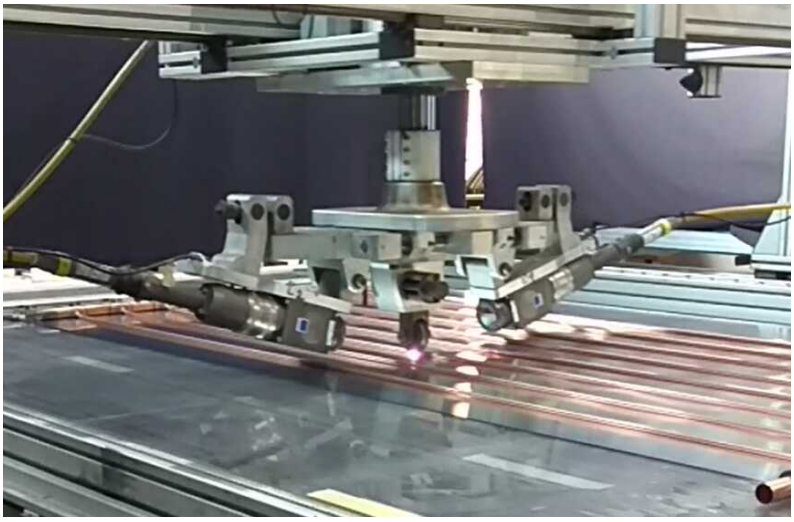
Ο κάθε τύπος προϊόντος απαιτεί κατάλληλη ρύθμιση των συνθηκών λειτουργίας της μηχανής laser αλλά και των ιδιοσυσκευών συγκράτησης πάνω στις τράπεζες («σετάρισμα»). Συγκεκριμένα, τα βήματα της διαδικασίας σεταρίσματος καθορίζονται από κάποιες βασικές παραμέτρους που προκύπτουν από τις διαστάσεις κάθε προϊόντος (οι οποίες μπορεί να διαφέρουν ακόμα και για προϊόντα ίδιου τύπου). Αυτές είναι η απόσταση κέντρο-κέντρο των header, το μήκος ανοιχτών εξόδων header (από την άκρη του φύλλου), οι θέσεις των κλειστών εξόδων (αν υπάρχουν), η απόσταση του πρώτου riser από την μία άκρη του φύλλου, το βήμα των riser (και κατ΄ επέκταση των φινακιών, αν υπάρχουν), και το πλάτος των φινακιών. Η γενικευμένη διαδικασία σεταρίσματος (όπως εφαρμόζεται έως τώρα), από την οποία μπορεί να προκύψει η κατάλληλη για κάθε προϊόν εισάγοντας τις παραμέτρους του, έχει καταγραφεί με τη μέθοδο της ιεραρχικής αξιολόγησης καθηκόντων (Ι.Α.Κ.).

Εξαιτίας της συχνής εναλλαγής μεταξύ διαφορετικών προϊόντων κρίνεται σημαντικός, στο σύνολό του, ο χρόνος ρύθμισης της μηχανής ενώ η πιθανότητα σφάλματος δεν μπορεί να αγνοηθεί. Η αυτοματοποίηση αυτής της διαδικασίας μπορεί να μειώσει σημαντικά τον χρόνο σεταρίσματος και την πιθανότητα σφάλματος διατηρώντας ταυτόχρονα την ευελιξία του υπάρχοντος συστήματος παραγωγής. Κάτι τέτοιο μπορεί να επιτευχθεί με μία αυτόματη ιδιοσυσκευή συγκράτησης η οποία λαμβάνει τα στοιχεία του προϊόντος μέσω υπολογιστή και διαμορφώνεται κατάλληλα χωρίς να απαιτείται παρέμβαση από τον χειριστή. Στα κεφάλαια που ακολουθούν περιγράφεται η διαδικασία σχεδιασμού και δοκιμής μίας τέτοιας ιδιοσυσκευής με στόχο την τελική ενσωμάτωσή και χρήση της στην υπάρχουσα μηχανή συγκόλλησης με laser.

## 3.2 Περιγραφή υπάρχουσας μεθόδου προετοιμασίας της μηχανής

### 3.2.1 Τύποι συλλεκτών

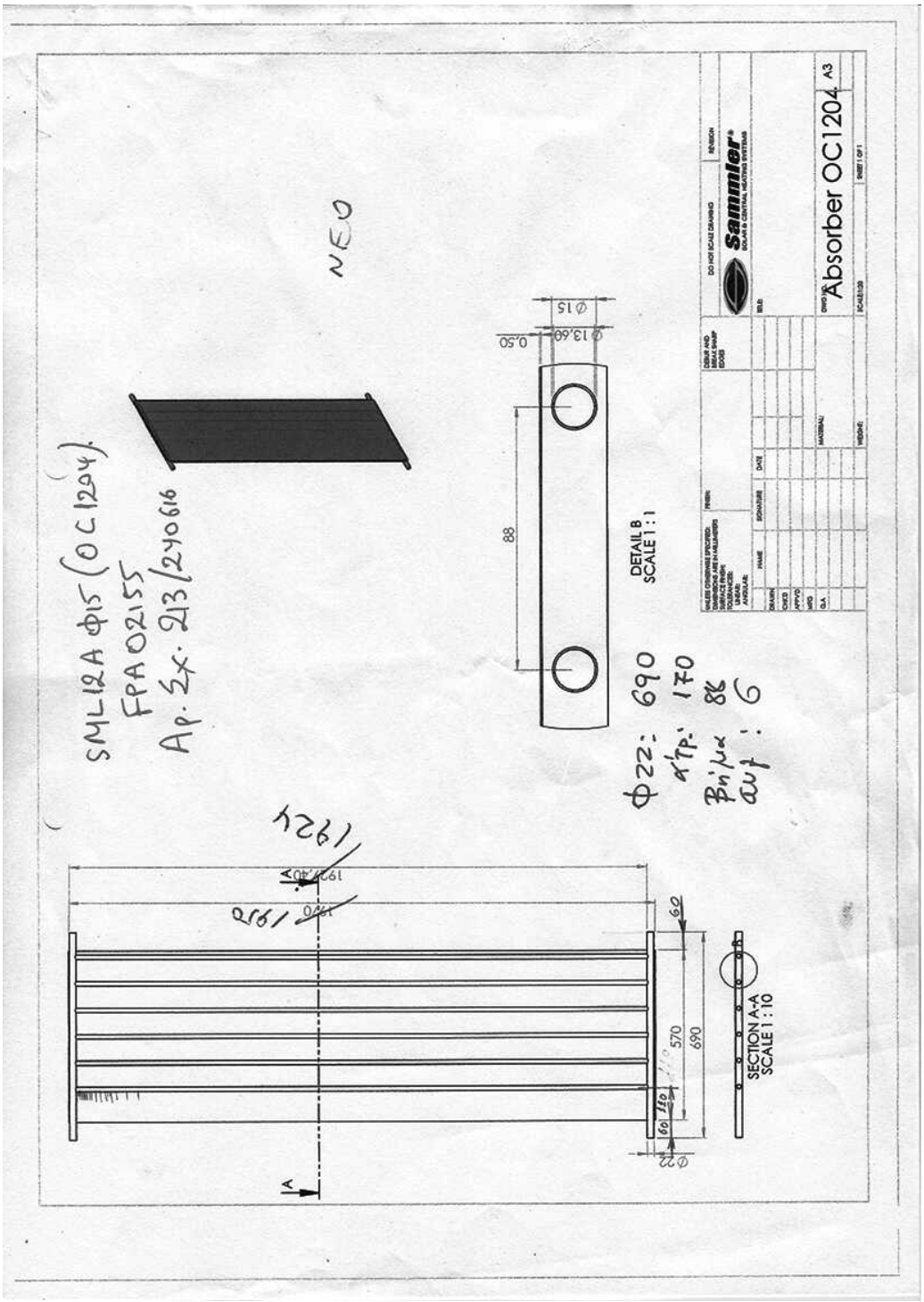
Το προϊόν, σε κάθε περίπτωση, αποτελείται από ένα πλέγμα χάλκινων σωλήνων συγκολλημένων στην πίσω πλευρά ενός βαμμένου απορροφητικού φύλλου από αλουμίνιο. Στις δύο εκατέρωθεν πλευρές του πλέγματος βρίσκονται τα κολεκτέρ ή headers (συνήθως ⌀22) οι οποίοι συνδέονται με τους μικρότερους σε διάμετρο riser (συνήθως ⌀8). Από τις τέσσερις εξόδους των header ενδέχεται να είναι κλειστή η μία από κάθε πλευρά (βλ. σχήμα 4). Πριν συγκολληθεί, το πλέγμα καμπυλώνεται ελαφρώς, με στράντζα, στα σημεία ένωσης των riser με τους header (βλ. λεπτομέρεια στο σχήμα 4) έτσι ώστε να μπορούν να έρθουν σε επαφή οι risers με μία επίπεδη επιφάνεια. Η ραφές της συγκόλλησης γίνονται πάντα κατά μήκος των riser όποιος και αν είναι ο προσανατολισμός τους (βλ. εικόνα 1).



Εικόνα 1: Συγκόλληση με laser κατά μήκος ενός riser σε κατακόρυφο συλλέκτη.

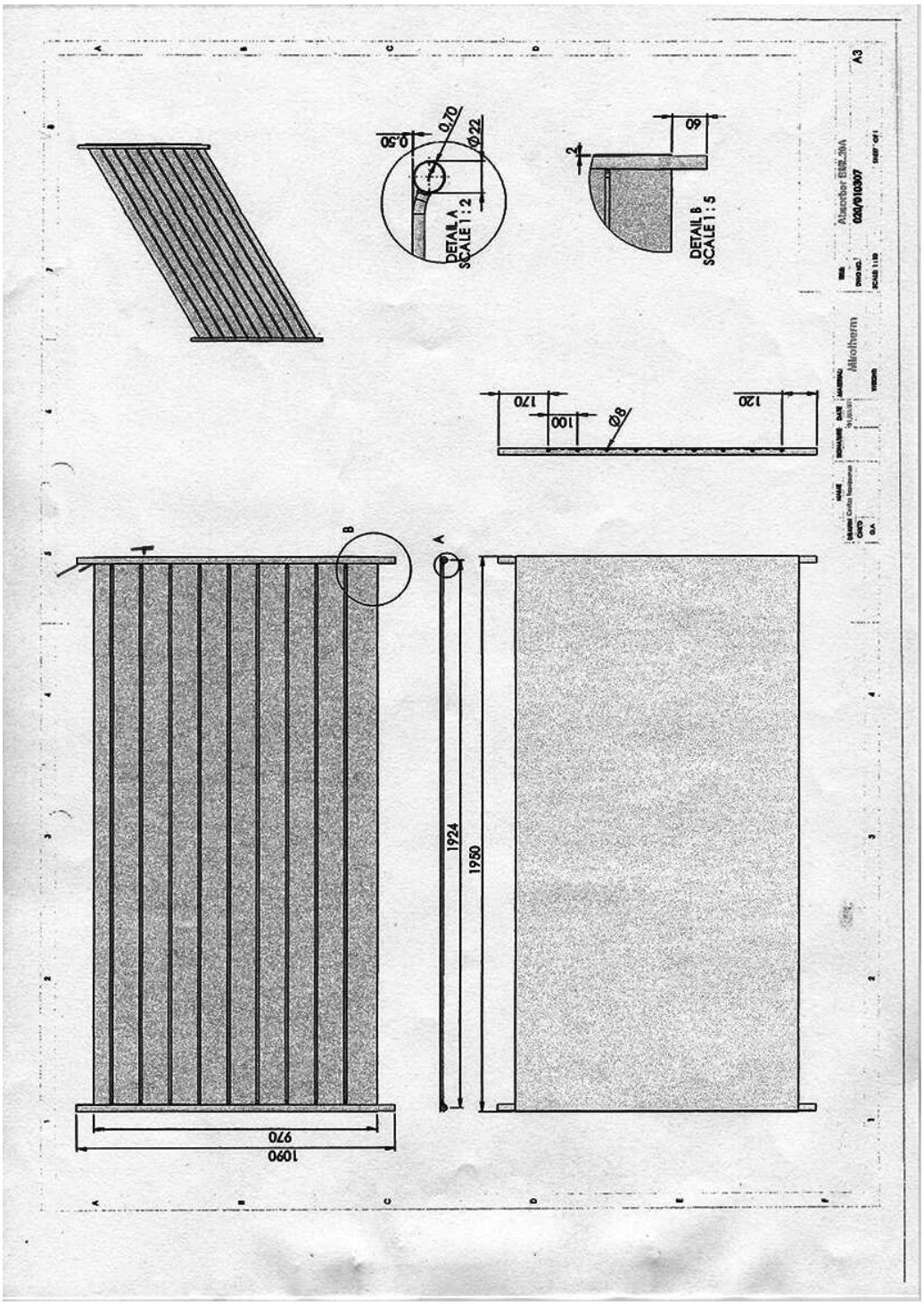
Οι πιθανές παραλλαγές στη γεωμετρία αφορούν τη διάταξη των riser και την τοποθέτηση των header κατά το μήκος ή το πλάτος. Διακρίνονται οι παρακάτω τύποι προϊόντων:

- *Σχάρα κατακόρυφη*: (βλ. σχήματα 1, 2) οι riser είναι τοποθετημένοι παράλληλα μεταξύ τους κατά το μήκος του φύλλου και, επειδή η μεγαλύτερη διάσταση είναι αυτή κατά το μήκος των riser, το προϊόν χαρακτηρίζεται ως «κατακόρυφο».
- *Σχάρα οριζόντια*: (βλ. σχήματα 3) οι riser είναι τοποθετημένοι με τον ίδιο ακριβώς τρόπο με πριν αλλά κατά το πλάτος του φύλλου σε αυτή την περίπτωση, και για αυτό το προϊόν χαρακτηρίζεται ως «οριζόντιο».
- *Με λωρίδες («φινάκια»)*: (βλ. σχήματα 4) το προϊόν είναι μορφολογικά και λειτουργικά ίδιο με αυτό των προηγούμενων δύο περιπτώσεων (συνηθέστερος είναι ο κατακόρυφος τύπος) με τη διαφορά ότι δε χρησιμοποιήθηκε ένα απορροφητικό φύλλο αλλά πολλές λωρίδες τοποθετημένες δίπλα-δίπλα. Συνήθως το πλάτος και η επικάλυψή τους επιλέγονται έτσι ώστε να υπάρχει μία λωρίδα κάτω από κάθε riser. Αυτή η περίπτωση προϊόντος είναι σπανιότερη.
- *Με μαίανδρο*: (βλ. σχήματα 5) σε αυτήν την περίπτωση ένας μόνο riser συνδέει τους δύο header, σε μορφή μαιάνδρου. Και αυτή η περίπτωση προϊόντος είναι σπανιότερη.



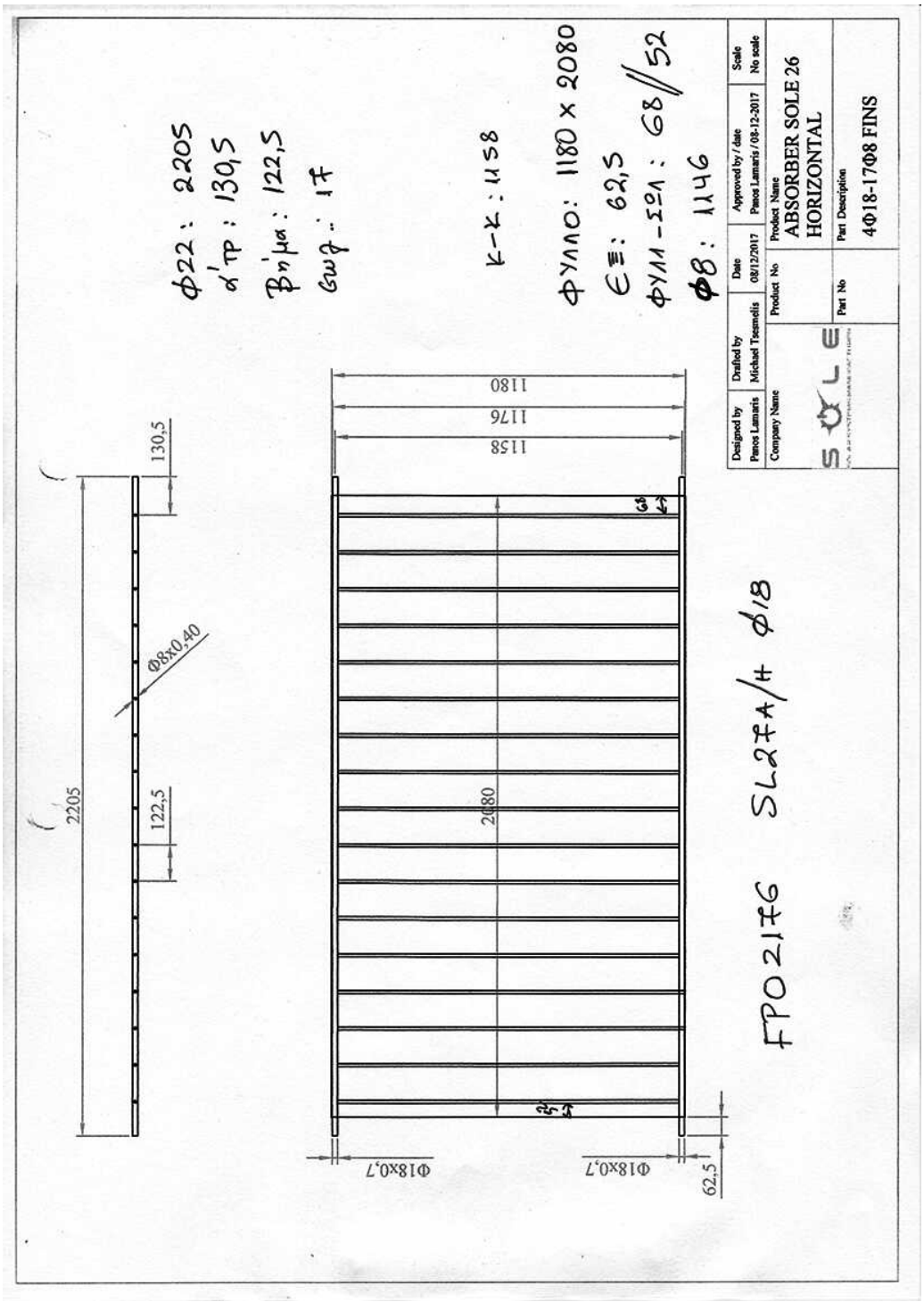

Σχήμα 1: Ηλιακός συλλέκτης με «κατακόρυφη» σχάρα riser.

- *Ανοιχτές και οι τέσσερις έξοδοι των header.*
- *Απόσταση κέντρο-κέντρο*: 1924 mm
- *Μήκος εξόδων header*: 60 mm
- *Απόσταση α΄ σωλήνα από φύλλο*: 110 mm
- *Βήμα*: 88 mm



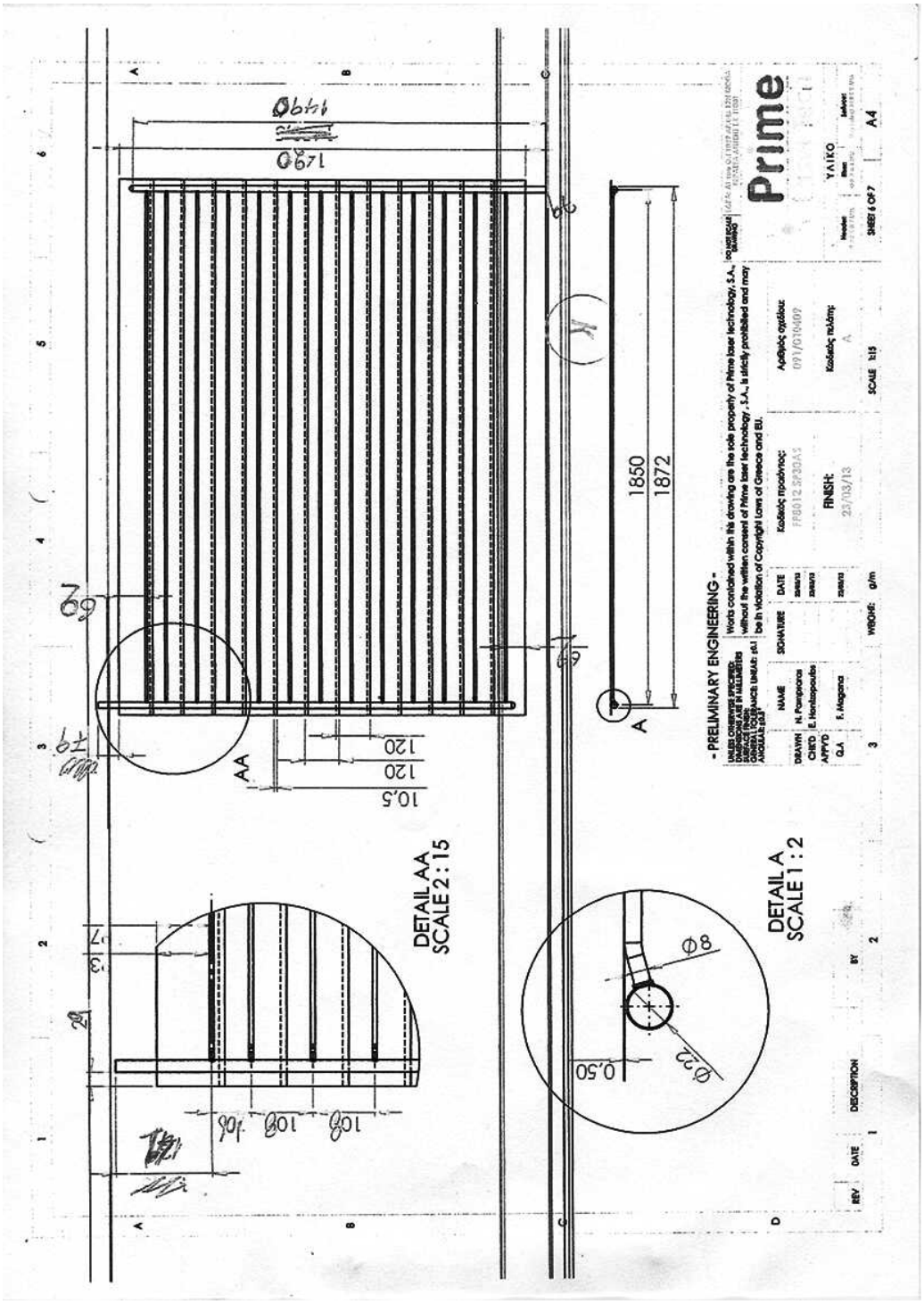

Σχήμα 2: Ηλιακός συλλέκτης με «κατακόρυφη» σχάρα riser.

- *Ανοιχτές και οι τέσσερις έξοδοι των header.*
- *Απόσταση κέντρο-κέντρο*: 1924 mm
- *Μήκος εξόδων header*: 60 mm
- *Απόσταση α΄ σωλήνα από φύλλο*: 170 mm
- *Απόσταση τελευταίου σωλήνα από φύλλο*: 120 mm
- *Βήμα*: 100 mm



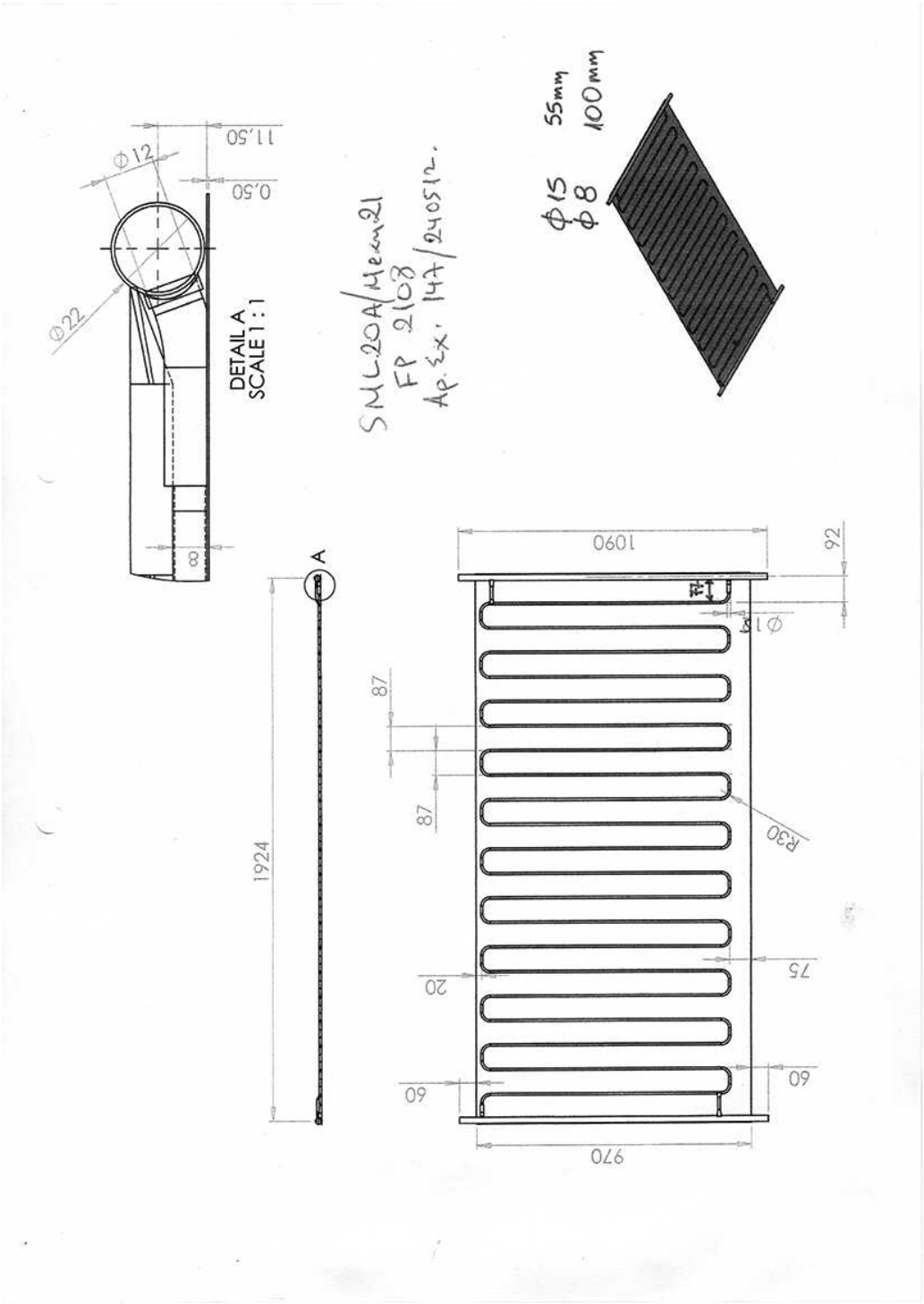

Σχήμα 3: Ηλιακός συλλέκτης με «οριζόντια» σχάρα riser.

- *Ανοιχτές και οι τέσσερις έξοδοι των header.*
- *Απόσταση κέντρο-κέντρο*: 1158 mm
- *Μήκος εξόδων header*: 62.5 mm
- *Απόσταση α΄ σωλήνα από φύλλο*: 68 mm
- *Απόσταση τελευταίου σωλήνα από φύλλο*: 52 mm
- *Βήμα*: 122.5 mm




Σχήμα 4: Ηλιακός συλλέκτης με «φινάκια».

- *Μία έξοδος σε κάθε header ανοιχτή.*
- *Απόσταση κέντρο-κέντρο*: 1850 mm
- *Μήκος εξόδων header*: 79 mm
- *Απόσταση α΄ σωλήνα από φύλλο*: 62 mm
- *Βήμα*: 108 mm
- *Πλάτος λωρίδων*: 120 mm
- *Επικάλυψη λωρίδων*: 10.5 mm




Σχήμα 5: Ηλιακός συλλέκτης με μαίανδρο riser.

- *Ανοιχτές και οι τέσσερις έξοδοι των header.*
- *Απόσταση κέντρο-κέντρο*: 1924 mm
- *Μήκος εξόδων header*: 60 mm
- *Απόσταση α΄ σωλήνα από φύλλο*: 75 mm
- *Απόσταση τελευταίου σωλήνα από φύλλο*: 20 mm
- *Βήμα*: 87 mm

Οι βασικές διαστάσεις που καθορίζουν την ρύθμιση της μηχανής συγκόλλησης με laser, και δίνονται στους χειριστές με την εντολή εργασίας (βλ.σχήμα 6) είναι οι εξής:

- *Απόσταση κέντρο-κέντρο*: (βλ. σχήματα 1-5) είναι η απόσταση μεταξύ των κέντρων των δύο header, δεν ταυτίζεται απαραίτητα με το μήκος του φύλλου, αλλά με βάση αυτή γίνεται η σχετική τοποθέτηση με το φύλλο στην μία διεύθυνση (συνήθως την κατακόρυφη), ενώ είναι απαραίτητη για το πρόγραμμα του laser ώστε να προσδιοριστεί το μήκος που συγκολλάται.
- *Μήκος εξόδων header*: (βλ. σχ. 1-5) είναι το μήκος των άκρων των header που προεξέχουν από τη μία πλευρά του φύλλου, χρησιμοποιείται για την σχετική τοποθέτηση του φύλλου και του πλέγματος σωλήνων τοποθέτηση στην μία διεύθυνση (συνήθως την οριζόντια).
- *Απόσταση α΄ σωλήνα (riser) από φύλλο*: (βλ. σχ. 1-5) είναι η απόσταση του πρώτου riser από την μία πλευρά του φύλλου, την ίδια πλευρά που δόθηκε και η προηγούμενη διάσταση, και χρησιμοποιείται μαζί με εκείνη για την σχετική τοποθέτηση του φύλλου και των σωλήνων στην μία διεύθυνση (συνήθως την οριζόντια).
- *Βήμα*: (βλ. σχ. 1-5) είναι η απόσταση μεταξύ των riser, εισάγεται στο πρόγραμμα του laser.
- *Πλάτος λωρίδων*: χρησιμοποιείται στην τοποθέτηση των φύλλων/λωρίδων στην περίπτωση μόνο του προϊόντος με λωρίδες (βλ. σχήμα 4).
- *Αλληλοεπικάλυψη λωρίδων*: χρησιμοποιείται μαζί με την προηγούμενη διάσταση στην τοποθέτηση των φύλλων/λωρίδων, και πάλι μόνο στην περίπτωση του προϊόντος με λωρίδες (βλ. σχήμα 4).

**ΕΝΤΟΛΗ ΕΡΓΑΣΙΑΣ 7 : Παραγωγή Απορροφητή Grid**

| | | | | |
|---|---|---|---|---|
| ΕΝΤΟΛΗ ΠΑΡΑΓΩΓΗΣ | | 18101105515 / 4861 | | |
| ΤΥΠΟΣ ΠΡΟΪΟΝΤΟΣ | | FPA02155SML12-115224E | | |
| ΤΕΜΑΧΙΑ | | 68 | | |
| ΠΟΙΟΤΗΤΑ ΕΠΙΦΑΝΕΙΑΣ | | Mirotherm | | |
| ΚΩΔΙΚΟΣ ΦΥΛΛΟΥ | | RMS02A501170 | | |
| ΔΙΑΣΤΑΣΕΙΣ ΦΥΛΛΟΥ | | 570 x 1.950 | | |
| **ΠΛΑΤΟΣ ΑΠΟΡΡΟΦΗΤΗ** | | ΒΗΜΑ | 88 | |
| COIL No | | SERIAL No | | |
| COIL No | | SERIAL No | | |
| ΑΠΟΣΤΑΣΗ ΚΕΝΤΡΟ-ΚΕΝΤΡΟ | | 1924 | | |
| ΜΗΚΟΣ ΕΞΟΔΩΝ HEADER | | 60 | | |
| ΑΠΟΣΤΑΣΗ Α' ΣΩΛΗΝΑ ΑΠΟ ΦΥΛΛΟ | | 110 | | |
| ΔΙΑΣΤΑΣΙΟΛΟΓΙΚΟΣ ΕΛΕΓΧΟΣ | ΑΡΧΗ | ΜΕΣΗ | ΤΕΛΟΣ | |
| | | | | |
| ΠΑΡΑΤΗΡΗΣΕΙΣ / ΣΧΟΛΙΑ ΑΠΟ ΧΕΙΡΙΣΤΕΣ: | | | | |

Page 4 of 5

Σχήμα 6: Η εντολή εργασίας που δίνεται στους χειριστές της μηχανής συγκόλλησης Laser για την παραγωγή προϊόντων με «κατακόρυφη» σχάρα riser. Οι χρήσιμες διαστάσεις είναι η απόσταση κέντρο-κέντρο, το μήκος εξόδων header, η απόσταση α΄ σωλήνα από το φύλλο, και το βήμα.

### 3.2.2 Ρύθμιση της μηχανής από τους χειριστές

Σε κάθε αλλαγή παρτίδας απαιτείται ρύθμιση της μηχανής. Αυτή περιλαμβάνει την αφαίρεση ή τοποθέτηση των κατάλληλων ιδιοσυσκευών στήριξης και την ρύθμιση του προγράμματος του laser. Κατά την τοποθέτηση των ιδιοσυσκευών λαμβάνονται υπόψη όλες οι προαναφερθείσες διαστάσεις κάθε τύπου προϊόντος, ενώ για τη τροποποίηση του προγράμματος του laser χρειάζονται, συνήθως, το πλήθος και το βήμα των riser, και η απόσταση κέντρο-κέντρο. Η διαδικασία που ακολουθείται είναι η εξής:

- Συγκεντρώνονται όλες οι απαραίτητες επιπλέον ιδιοσυσκευές και τα εργαλεία για την εγκατάστασή τους, όπως κατσαβίδια. Εάν το προϊόν έχει λωρίδες αντί για φύλλο τότε χρειάζονται και τα αντίστοιχα μέτρα για την τοποθέτηση των οριζόντιων οδηγών. Εάν το πλέγμα των σωλήνων είναι σε μορφή μαιάνδρου τότε είναι απαραίτητα και τα δύο ξύλινα πλαίσια που περιβάλλουν τους riser δεξιά και αριστερά.
- Στη συνέχεια μεταφέρονται οι υπάρχουσες ιδιοσυσκευές (όσες χρειάζεται) και των δύο τραπεζών σε νέες θέσεις.
  - Η μπάρα στην πάνω πλευρά της τράπεζας μεταφέρεται, μαζί με ό,τι είναι στερεωμένο πάνω της, έτσι ώστε η απόσταση μεταξύ των οριζόντιων οδηγών της και των απέναντι να είναι ίση με το μήκος του φύλλου ή των λωρίδων. Για να γίνει αυτό τοποθετείται πρώτα ένα φύλλο ή μία λωρίδα στην επιθυμητή θέση πάνω στην τράπεζα.
  - Εάν το πλάτος του νέου φύλλου ή όλων των λωρίδων είναι αρκετά μεγάλο τότε ο δεξιά οδηγός μεταφέρεται ώστε να ακουμπά στη δεξιά πλευρά του φύλλου ή της δεξιάς λωρίδας, και κολλάται με ταινία.
  - Στην περίπτωση των λωρίδων χρειάζεται η μετακίνηση όλων των οριζόντιων οδηγών, και των δύο πλευρών, ώστε να ισαπέχουν μεταξύ τους όσο το πλάτος ενός φύλλου. Συγκεκριμένα, βιδώνονται ένας-ένας από δεξιά προς τα αριστερά, σε ίσες αποστάσεις χρησιμοποιώντας ένα μέτρο με το κατάλληλο μήκος. Αυτό το μήκος είναι ίσο με το πλάτος ενός φύλλου μείον την αλληλοεπικάλυψη μεταξύ δύο λωρίδων.
  - Ύστερα τοποθετείται πάνω από το φύλλο το πλέγμα σωληνών στην κατάλληλη θέση. Αυτή προσδιορίζεται με βάση το μήκος εξόδων των header και την απόσταση του α΄ riser από την δεξιά πλευρά του φύλλου. Αφού επιβεβαιωθεί ότι οι riser δεν είναι καμπυλωμένοι τοποθετούνται «στοπ» στις δύο δεξιά εξόδους των header. Τα «στοπ» κολλώνται με αυτοκόλλητο πάνω στην τράπεζα και στηρίζονται στην πίσω πλευρά τους με έλασμα βιδωμένο στη δεξιά πλευρά της τράπεζας.
  - Οι τέσσερις γάντζοι (δύο σε κάθε πλευρά) πιάνονται στους header στις κατάλληλες θέσεις και βιδώνονται.
  - Ομοίως, τοποθετούνται και τα ελάσματα που συγκρατούν στη μέση κάθε header, ρυθμίζεται το ύψος τους προσθέτοντας ή αφαιρώντας ροδέλες από κάτω τους, και βιδώνονται μέχρι να είναι σφιχτά.
  - Στην περίπτωση του "οριζόντιου" προϊόντος τα παραπάνω πραγματοποιούνται σε περιστροφή 90°. Δηλαδή χρησιμοποιείται νέα μπάρα κάθετη στην προηγούμενη, οι γάντζοι και τα ελάσματα βιδώνονται πάνω σε αυτή και στη δεξιά πλευρά της τράπεζας, τα «στοπ» κολλώνται στην κάτω πλευρά της τράπεζας κ.ο.κ.

- Αφού εξασφαλισθεί ότι τα δύο κομμάτια είναι καλά συγκρατημένα πραγματοποιείται η ρύθμιση του προγράμματος του laser. Επιλέγεται από τον υπολογιστή μία περίπτωση αντίστοιχου προϊόντος (αν δεν υπάρχει ήδη το συγκεκριμένο) και εισάγονται το πλήθος και το βήμα των riser, και η απόσταση κέντρο-κέντρο. Εισάγονται δηλαδή τα ευθύγραμμα τμήματα τα οποία θα συγκολληθούν.
- Στη συνέχεια προσδιορίζεται το offset που πρέπει να δοθεί ώστε να κατέβει και να ακουμπήσει ομαλά το ροδάκι της βάσης του laser πάνω στον πρώτο riser. Εκτελείται βήμα-βήμα το πρόγραμμα μέχρι το κατέβασμα της βάσης και με παρατήρηση και δοκιμή και σφάλμα προσδιορίζεται το offset. Εφόσον το βήμα είναι σταθερό δεν υπάρχει πρόβλημα στους επόμενους riser.

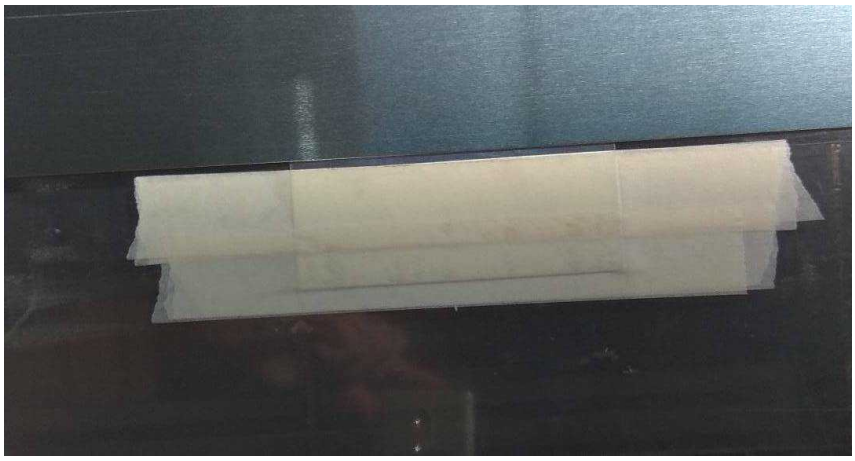
Αξίζει να σημειωθεί ότι δεν εκτελείται κάθε βήμα της παραπάνω διαδικασίας σε κάθε αλλαγή παρτίδας. Οι συνηθέστερες ενέργειες είναι η μεταφορά της μπάρας στο μήκος του φύλλου, η μετακίνηση των γάντζων και η ρύθμιση του προγράμματος του laser. Οι περιπτώσεις των προϊόντων με λωρίδες ή μαίανδρο είναι σπανιότερες από αυτές των προϊόντων με σχάρα σωληνών πάνω σε φύλλο (με συχνότητα όχι μεγαλύτερη από μία ανά εβδομάδα).

Εικόνα 2: Οδηγοί φύλλων (φινακιών) στην κάτω πλευρά της τράπεζας.

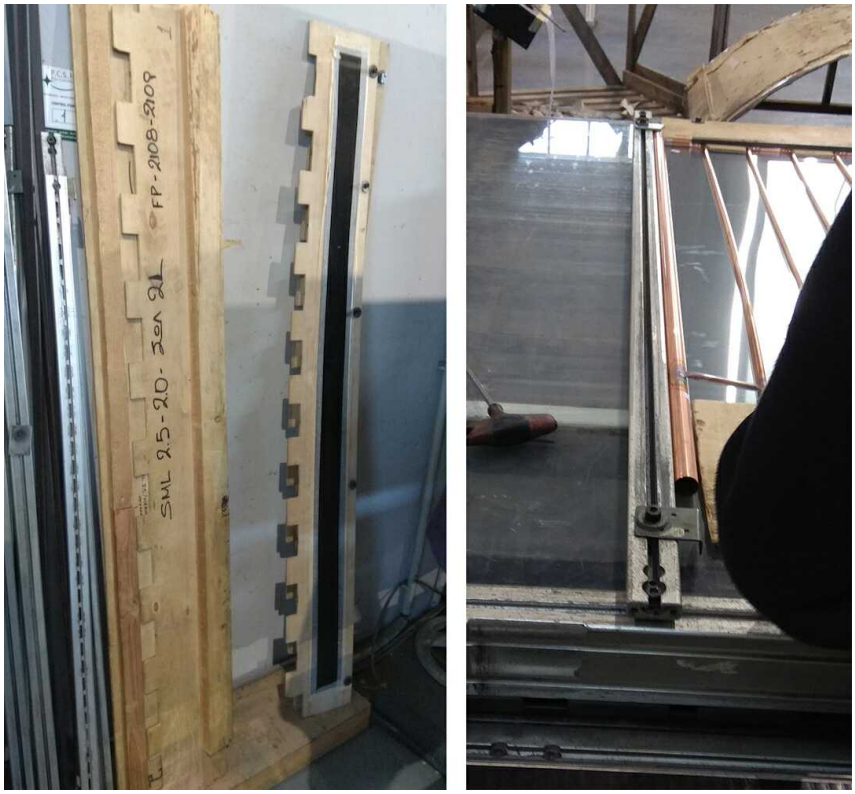
Εικόνα 3: Συγκρατητής-άγκιστρο στην κάτω πλευρά της τράπεζας.

Εικόνα 4: Στοπ τοποθετημένα στις εξόδους των header και πλευρικά τους, και από πάνω οι συγκρατητές ελάσματα σε προϊόν με μαίανδρο (αριστερή εικόνα: πάνω πλευρά τράπεζας, δεξιά εικόνα: κάτω πλευρά τράπεζας).

Εικόνα 5: Στοπ σε έξοδο header και ο συγκρατητής του βιδωμένος στο προφίλ αλουμινίου της δεξιάς πλευράς της τράπεζας.

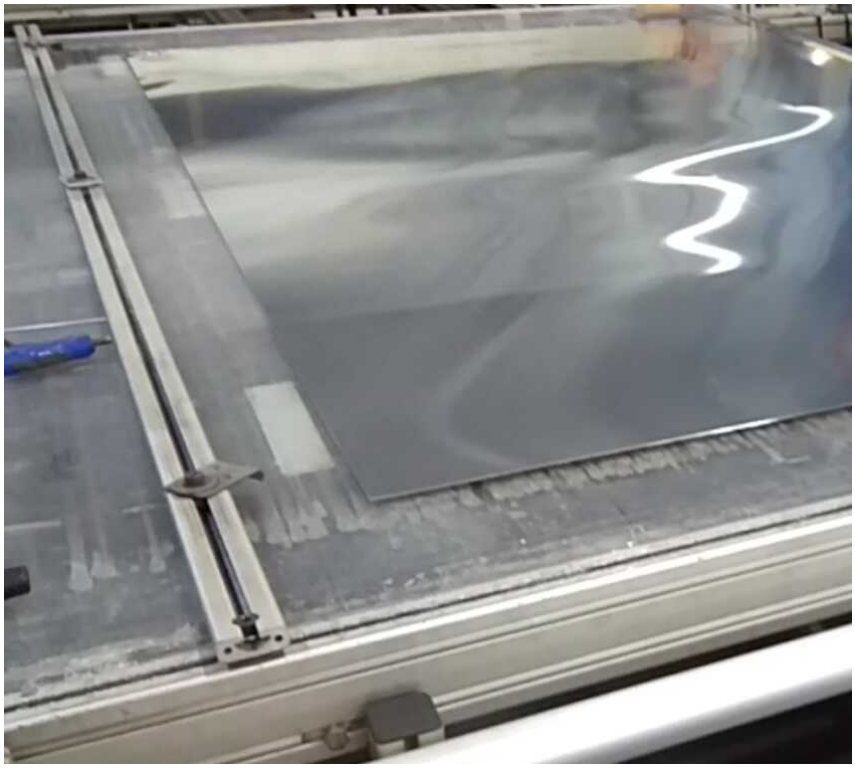


Εικόνα 6: Στοπ (τεμάχιο φύλλου) κολλημένο στη τράπεζα με χαρτοταινία.



Εικόνα 7: Τα ξύλινα πλαίσια και η μπάρα που τοποθετείται στη μία πλευρά της τράπεζας για το προϊόν με μαίανδρο.

Εικόνα 8: Μέτρα διαφόρων βημάτων που χρησιμοποιούνται στην τοποθέτηση των οδηγών των φινακιών.

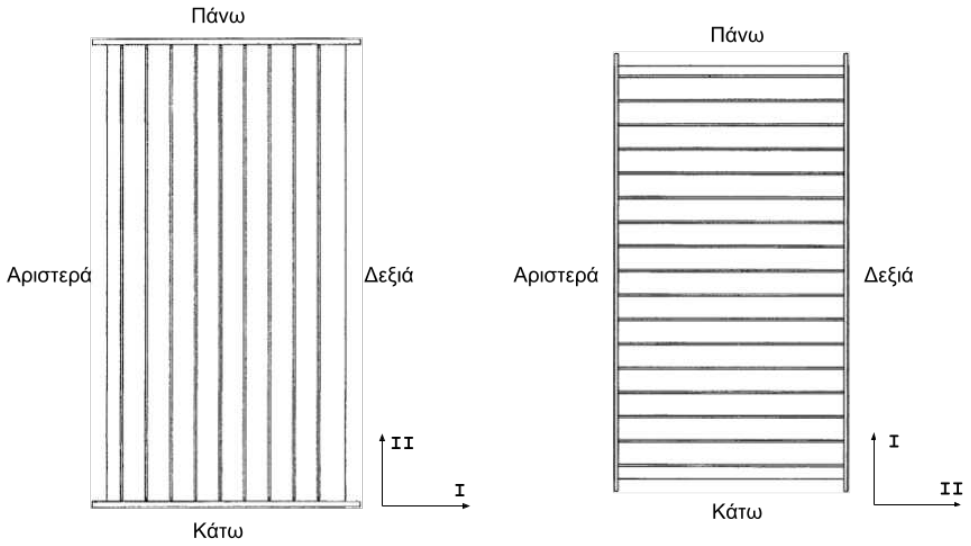


Εικόνα 9: Μπάρα που τοποθετείται κατά τη μεγάλη διάσταση της τράπεζας για το οριζόντιο προϊόν.

## 3.3 Υποστήριξη της υπάρχουσας διαδικασίας μέσω βελτίωσης της εντολής εργασίας

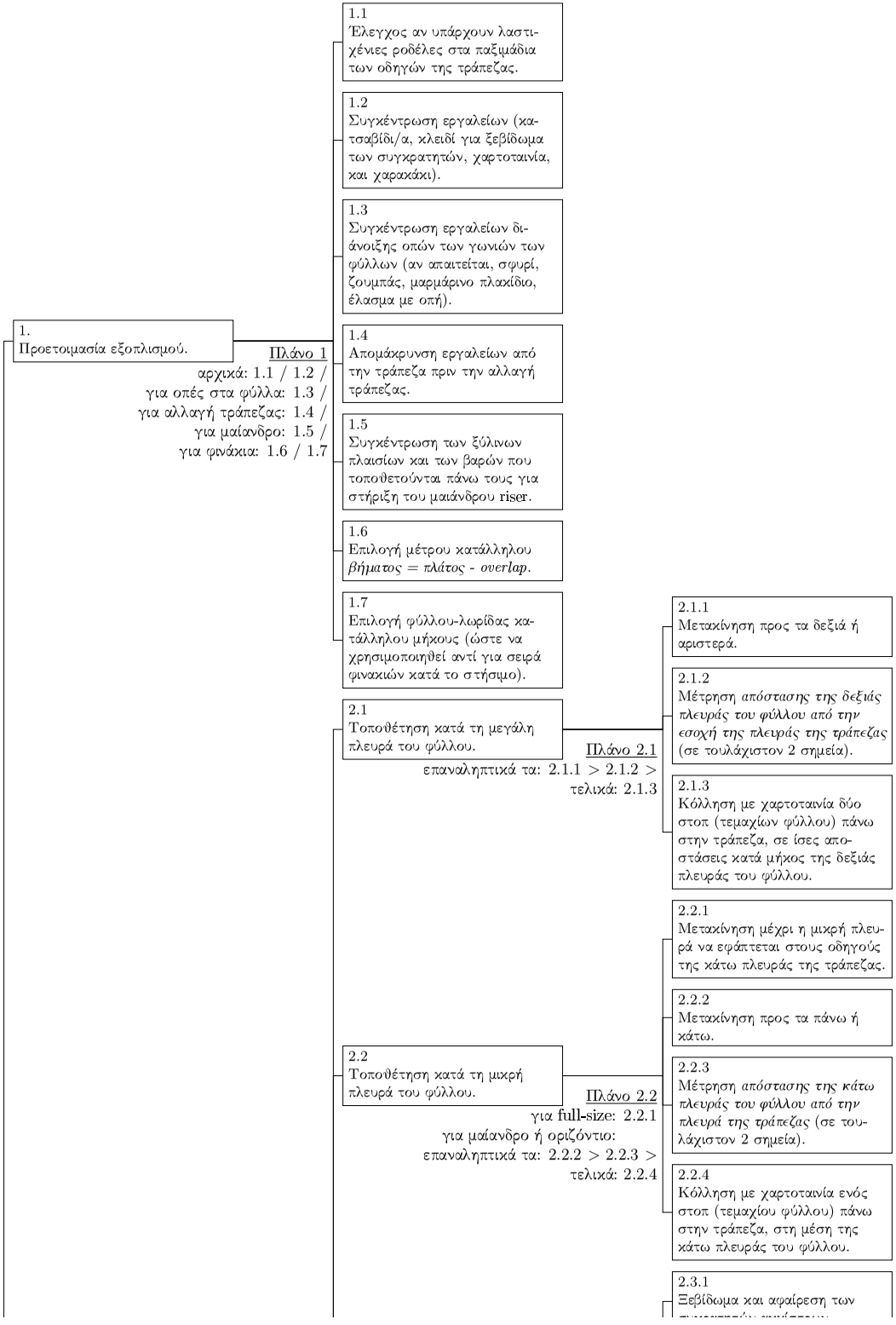
### 3.3.1 Ιεραρχική ανάλυση καθηκόντων της διαδικασίας προετοιμασίας

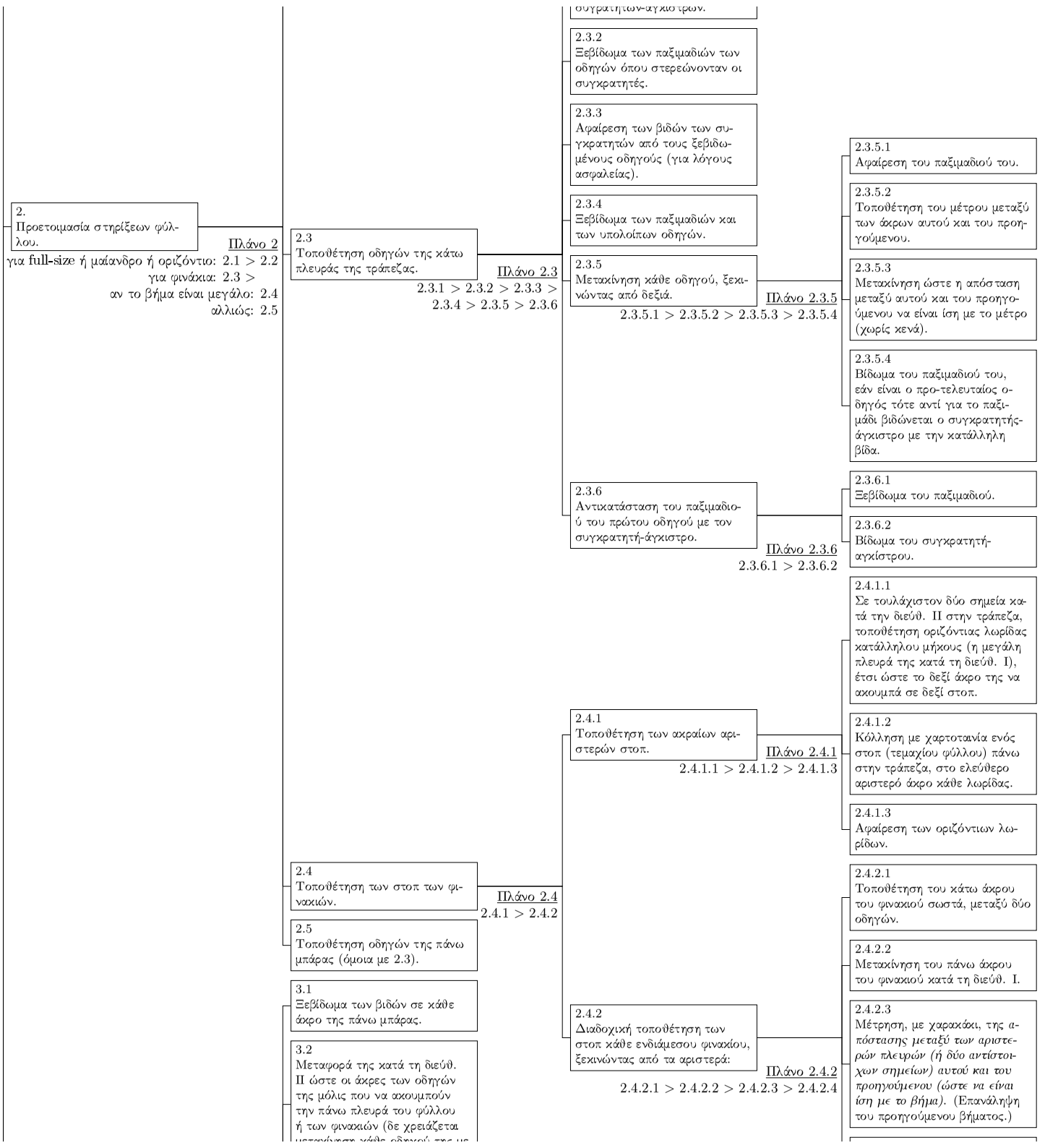
Από τις παρατηρήσεις που συλλέχθηκαν ύστερα από παρακολούθηση καταγράφηκε η σειρά των βημάτων ρύθμισης για κάθε προϊόν και χωρίστηκαν σε καθήκοντα και υπο-καθήκοντα, σε επίπεδα. Παρατηρήθηκε ότι σε κάθε περίπτωση τα καθήκοντα των ανώτερων επιπέδων παραμένουν τα ίδια ενώ παραλλάσσονται τα καθήκοντα των κατώτερων επιπέδων. Συνεπώς συγκεντρώθηκαν όλα τα βήματα σε μία διαδικασία η οποία εκτελείται σε παραλλαγές αναλόγως την περίπτωση προϊόντος και τις παραμέτρους που αναφέρθηκαν προηγουμένως. Το προκύπτον διάγραμμα φαίνεται στην επόμενη σελίδα. Σε κάθε διακλάδωση (υπο-καθήκοντα ενός καθήκοντος) είναι σημειωμένη η απαραίτητη πορεία εκτέλεσης και διευκρινίζονται οι περιπτώσεις στις οποίες απαιτούνται συγκεκριμένα υπο-καθήκοντα. Με το σύμβολο "`>`" μεταξύ δύο καθηκόντων υποδηλώνεται αυστηρή διαδοχή ενώ με το "`/`" χαλαρή διαδοχή, εναλλαγή, ή ακόμα και παράλληλη υλοποίησή τους. Με *πλάγια γράμματα* είναι σημειωμένες οι ενέργειες στις οποίες χρησιμοποιείται η τιμή κάποιας διάστασης κατά τη ρύθμιση. Οι χρησιμοποιούμενοι όροι για την τοπολογία πάνω στην τράπεζα φαίνονται στο σχήμα 7.

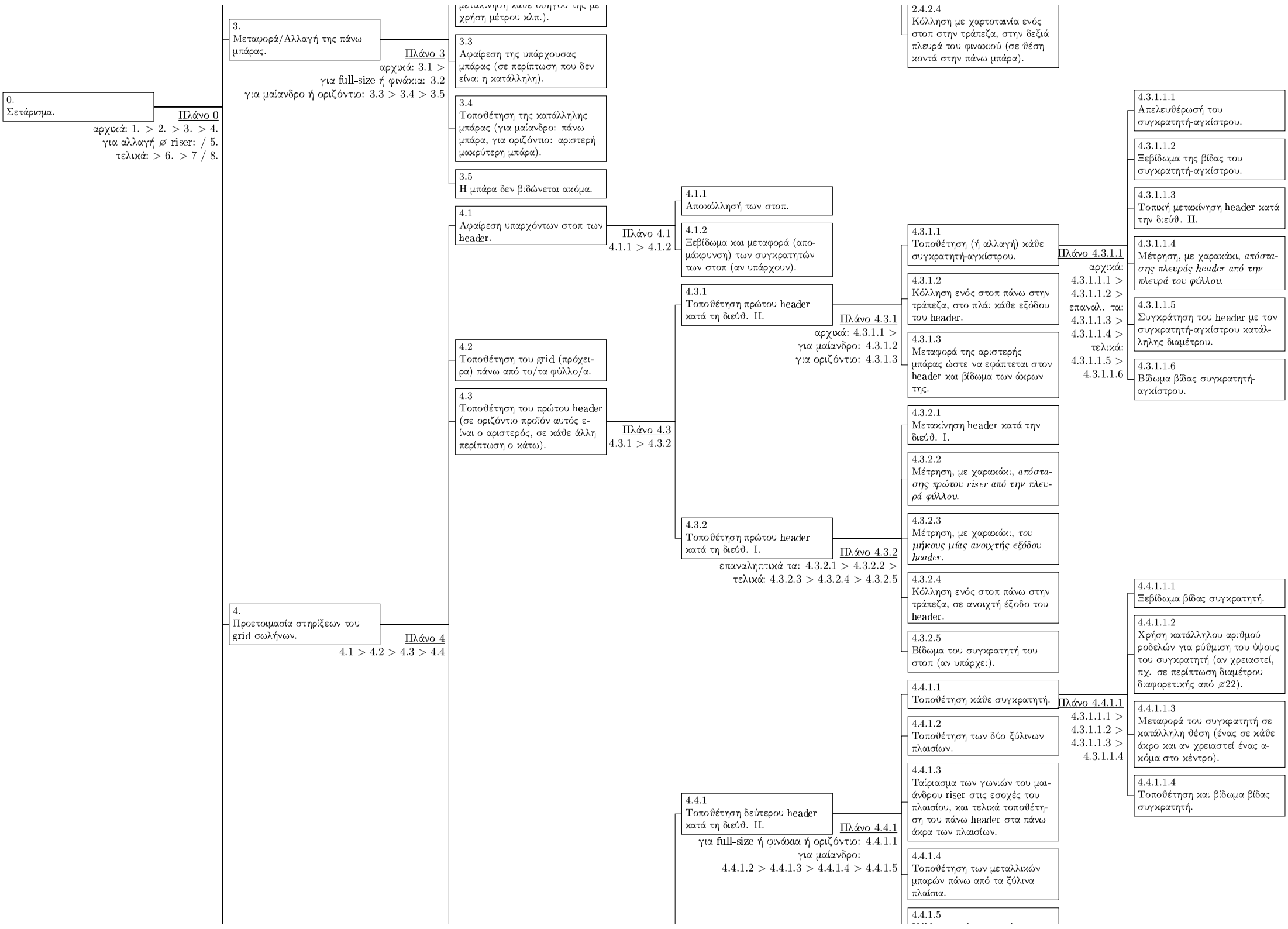


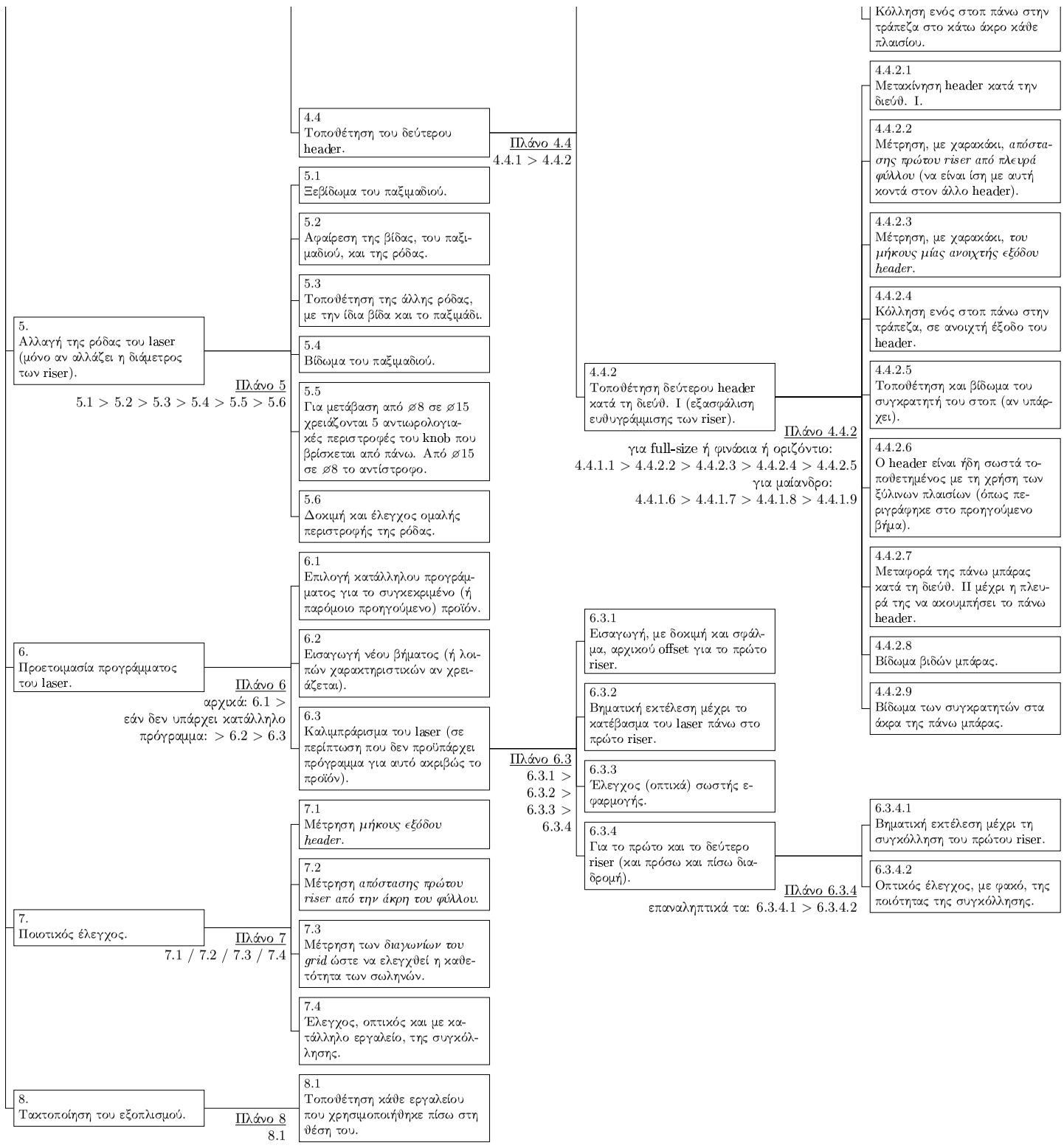

Σχήμα 7: Απεικόνιση της τοπολογίας όπως χρησιμοποιείται κατά την περιγραφή των βημάτων (αριστερά στην περίπτωση full-size και δεξιά στην περίπτωση οριζόντιου προϊόντος).
Διεύθυνση I: η αξονική διεύθυνση των header.
Διεύθυνση II: η ακτινική των header (κάθετη στην I).

Η διαδικασία αφορά την προετοιμασία της μίας τράπεζας (εκτελείται η ίδια και για την δεύτερη, με την παραλλαγή ενός βήματος). Ξεκινά με τη συγκέντρωση του απαραίτητου εξοπλισμού, θεωρώντας άδειες αρχικά τις τράπεζες, και τελειώνει μόλις μπορεί να αρχίσει η παραγωγή του πρώτου προϊόντος. Παρατηρείται ότι σε όσο πιο χαμηλά επίπεδα βρίσκεται μία σειρά καθηκόντων τόσο μικρότερη ευελιξία υπάρχει στην εναλλαγή της σειράς εκτέλεσης. Ο μέγιστος βαθμός ευελιξίας είναι εμφανής μόνο στα βασικά καθήκοντα των πάνω επιπέδων. Αυτό βέβαια δεν απαγορεύει από δύο χειριστές την παράλληλη εκτέλεση (μερικών) καθηκόντων μεταξύ ενός επιπέδου ή υπο-καθηκόντων μεταξύ επιπέδων (λχ. ο πρώτος ξεβιδώνει τις βίδες της πάνω μπάρας (3.1) ενώ ο δεύτερος μετακινεί το φύλλο κατά την αρχική τοποθέτησή του (2.1.1)).


1.
Προετοιμασία εξοπλισμού.
Πλάνο 1
αρχικά: 1.1 / 1.2 /
για οπές στα φύλλα: 1.3 /
για αλλαγή τράπεζας: 1.4 /
για μαίανδρο: 1.5 /
για φινάκια: 1.6 / 1.7
1.1
Έλεγχος αν υπάρχουν λαστιχένιες ροδέλες στα παξιμάδια των οδηγών της τράπεζας.
1.2
Συγκέντρωση εργαλείων (κατσαβίδι/α, κλειδί για ξεβίδωμα των συγκρατητών, χαρτοταινία, και χαρακάκι).
1.3
Συγκέντρωση εργαλείων διάνοιξης οπών των γωνιών των φύλλων (αν απαιτείται, σφυρί, ζουμπάς, μαρμάρινο πλακίδιο, έλασμα με οπή).
1.4
Απομάκρυνση εργαλείων από την τράπεζα πριν την αλλαγή τράπεζας.
1.5
Συγκέντρωση των ξύλινων πλαισίων και των βαρών που τοποθετούνται πάνω τους για στήριξη του μαιάνδρου riser.
1.6
Επιλογή μέτρου κατάλληλου *βήματος = πλάτος - overlap*.
1.7
Επιλογή φύλλου-λωρίδας κατάλληλου μήκους (ώστε να χρησιμοποιηθεί αντί για σειρά φινακιών κατά το στήσιμο).
2.1
Τοποθέτηση κατά τη μεγάλη πλευρά του φύλλου.
Πλάνο 2.1
επαναληπτικά τα: 2.1.1 > 2.1.2 >
τελικά: 2.1.3
2.1.1
Μετακίνηση προς τα δεξιά ή αριστερά.
2.1.2
Μέτρηση *απόστασης της δεξιάς πλευράς του φύλλου από την εσοχή της πλευράς της τράπεζας* (σε τουλάχιστον 2 σημεία).
2.1.3
Κόλληση με χαρτοταινία δύο στοπ (τεμαχίων φύλλου) πάνω στην τράπεζα, σε ίσες αποστάσεις κατά μήκος της δεξιάς πλευράς του φύλλου.
2.2
Τοποθέτηση κατά τη μικρή πλευρά του φύλλου.
Πλάνο 2.2
για full-size: 2.2.1
για μαίανδρο ή οριζόντιο:
επαναληπτικά τα: 2.2.2 > 2.2.3 >
τελικά: 2.2.4
2.2.1
Μετακίνηση μέχρι η μικρή πλευρά να εφάπτεται στους οδηγούς της κάτω πλευράς της τράπεζας.
2.2.2
Μετακίνηση προς τα πάνω ή κάτω.
2.2.3
Μέτρηση *απόστασης της κάτω πλευράς του φύλλου από την πλευρά της τράπεζας* (σε τουλάχιστον 2 σημεία).
2.2.4
Κόλληση με χαρτοταινία ενός στοπ (τεμαχίου φύλλου) πάνω στην τράπεζα, στη μέση της κάτω πλευράς του φύλλου.
2.3.1
Ξεβίδωμα και αφαίρεση των


2.
Προετοιμασία στηρίξεων φύλλου.
Πλάνο 2
για full-size ή μαϊανδρο ή οριζόντιο: 2.1 > 2.2
για φινάκια: 2.3 >
αν το βήμα είναι μεγάλο: 2.4
αλλιώς: 2.5
2.3
Τοποθέτηση οδηγών της κάτω πλευράς της τράπεζας.
Πλάνο 2.3
2.3.1 > 2.3.2 > 2.3.3 > 2.3.4 > 2.3.5 > 2.3.6
2.3.2
Ξεβίδωμα των παξιμαδιών των οδηγών όπου στερεώνονταν οι συγκρατητές.
2.3.3
Αφαίρεση των βιδών των συγκρατητών από τους ξεβιδωμένους οδηγούς (για λόγους ασφαλείας).
2.3.4
Ξεβίδωμα των παξιμαδιών και των υπολοίπων οδηγών.
2.3.5
Μετακίνηση κάθε οδηγού, ξεκινώντας από δεξιά.
Πλάνο 2.3.5
2.3.5.1 > 2.3.5.2 > 2.3.5.3 > 2.3.5.4
2.3.5.1
Αφαίρεση του παξιμαδιού του.
2.3.5.2
Τοποθέτηση του μέτρου μεταξύ των άκρων αυτού και του προηγούμενου.
2.3.5.3
Μετακίνηση ώστε η απόσταση μεταξύ αυτού και του προηγούμενου να είναι ίση με το μέτρο (χωρίς κενά).
2.3.5.4
Βίδωμα του παξιμαδιού του, εάν είναι ο προ-τελευταίος οδηγός τότε αντί για το παξιμάδι βιδώνεται ο συγκρατητής-άγκιστρο με την κατάλληλη βίδα.
2.3.6
Αντικατάσταση του παξιμαδιού του πρώτου οδηγού με τον συγκρατητή-άγκιστρο.
Πλάνο 2.3.6
2.3.6.1 > 2.3.6.2
2.3.6.1
Ξεβίδωμα του παξιμαδιού.
2.3.6.2
Βίδωμα του συγκρατητή-αγκίστρου.
2.4
Τοποθέτηση των στοπ των φινακιών.
Πλάνο 2.4
2.4.1 > 2.4.2
2.4.1
Τοποθέτηση των ακραίων αριστερών στοπ.
Πλάνο 2.4.1
2.4.1.1 > 2.4.1.2 > 2.4.1.3
2.4.1.1
Σε τουλάχιστον δύο σημεία κατά την διεύθ. ΙΙ στην τράπεζα, τοποθέτηση οριζόντιας λωρίδας κατάλληλου μήκους (η μεγάλη πλευρά της κατά τη διεύθ. Ι), έτσι ώστε το δεξί άκρο της να ακουμπά σε δεξί στοπ.
2.4.1.2
Κόλληση με χαρτοταινία ενός στοπ (τεμαχίου φύλλου) πάνω στην τράπεζα, στο ελεύθερο αριστερό άκρο κάθε λωρίδας.
2.4.1.3
Αφαίρεση των οριζόντιων λωρίδων.
2.4.2
Διαδοχική τοποθέτηση των στοπ κάθε ενδιάμεσου φινακίου, ξεκινώντας από τα αριστερά:
Πλάνο 2.4.2
2.4.2.1 > 2.4.2.2 > 2.4.2.3 > 2.4.2.4
2.4.2.1
Τοποθέτηση του κάτω άκρου του φινακιού σωστά, μεταξύ δύο οδηγών.
2.4.2.2
Μετακίνηση του πάνω άκρου του φινακιού κατά τη διεύθ. Ι.
2.4.2.3
Μέτρηση, με χαρακάκι, της *απόστασης μεταξύ των αριστερών πλευρών (ή δύο αντίστοιχων σημείων) αυτού και του προηγούμενου (ώστε να είναι ίση με το βήμα).* (Επανάληψη του προηγούμενου βήματος.)
2.5
Τοποθέτηση οδηγών της πάνω μπάρας (όμοια με 2.3).
3.1
Ξεβίδωμα των βιδών σε κάθε άκρο της πάνω μπάρας.
3.2
Μεταφορά της κατά τη διεύθ. ΙΙ ώστε οι άκρες των οδηγών της μόλις που να ακουμπούν την πάνω πλευρά του φύλλου ή των φινακιών (δε χρειάζεται


0.
Σετάρισμα.
Πλάνο 0
αρχικά: 1. > 2. > 3. > 4.
για αλλαγή ⌀ riser: / 5.
τελικά: > 6. > 7 / 8.
3.
Μεταφορά/Αλλαγή της πάνω μπάρας.
Πλάνο 3
αρχικά: 3.1 >
για full-size ή φινάκια: 3.2
για μαίανδρο ή οριζόντιο: 3.3 > 3.4 > 3.5
3.3
Αφαίρεση της υπάρχουσας μπάρας (σε περίπτωση που δεν είναι η κατάλληλη).
3.4
Τοποθέτηση της κατάλληλης μπάρας (για μαίανδρο: πάνω μπάρα, για οριζόντιο: αριστερή μακρύτερη μπάρα).
3.5
Η μπάρα δεν βιδώνεται ακόμα.
4.
Προετοιμασία στηρίξεων του grid σωλήνων.
Πλάνο 4
4.1 > 4.2 > 4.3 > 4.4
4.1
Αφαίρεση υπαρχόντων στοπ των header.
Πλάνο 4.1
4.1.1 > 4.1.2
4.1.1
Αποκόλλησή των στοπ.
4.1.2
Ξεβίδωμα και μεταφορά (απομάκρυνση) των συγκρατητών των στοπ (αν υπάρχουν).
4.2
Τοποθέτηση του grid (πρόχειρα) πάνω από το/τα φύλλο/α.
4.3
Τοποθέτηση του πρώτου header (σε οριζόντιο προϊόν αυτός είναι ο αριστερός, σε κάθε άλλη περίπτωση ο κάτω).
Πλάνο 4.3
4.3.1 > 4.3.2
4.3.1
Τοποθέτηση πρώτου header κατά τη διεύθ. ΙΙ.
Πλάνο 4.3.1
αρχικά: 4.3.1.1 >
για μαίανδρο: 4.3.1.2
για οριζόντιο: 4.3.1.3
4.3.1.1
Τοποθέτηση (ή αλλαγή) κάθε συγκρατητή-αγκίστρου.
Πλάνο 4.3.1.1
αρχικά:
4.3.1.1.1 >
4.3.1.1.2 >
επαναλ. τα:
4.3.1.1.3 >
4.3.1.1.4 >
τελικά:
4.3.1.1.5 >
4.3.1.1.6
4.3.1.1.1
Απελευθέρωσή του συγκρατητή-αγκίστρου.
4.3.1.1.2
Ξεβίδωμα της βίδας του συγκρατητή-αγκίστρου.
4.3.1.1.3
Τοπική μετακίνηση header κατά την διεύθ. ΙΙ.
4.3.1.1.4
Μέτρηση, με χαρακάκι, *απόστασης πλευράς header από την πλευρά του φύλλου.*
4.3.1.1.5
Συγκράτηση του header με τον συγκρατητή-αγκίστρου κατάλληλης διαμέτρου.
4.3.1.1.6
Βίδωμα βίδας συγκρατητή-αγκίστρου.
4.3.1.2
Κόλληση ενός στοπ πάνω στην τράπεζα, στο πλάι κάθε εξόδου του header.
4.3.1.3
Μεταφορά της αριστερής μπάρας ώστε να εφάπτεται στον header και βίδωμα των άκρων της.
4.3.2
Τοποθέτηση πρώτου header κατά τη διεύθ. Ι.
Πλάνο 4.3.2
επαναληπτικά τα: 4.3.2.1 > 4.3.2.2 >
τελικά: 4.3.2.3 > 4.3.2.4 > 4.3.2.5
4.3.2.1
Μετακίνηση header κατά την διεύθ. Ι.
4.3.2.2
Μέτρηση, με χαρακάκι, *απόστασης πρώτου riser από την πλευρά φύλλου.*
4.3.2.3
Μέτρηση, με χαρακάκι, *του μήκους μίας ανοιχτής εξόδου header.*
4.3.2.4
Κόλληση ενός στοπ πάνω στην τράπεζα, σε ανοιχτή έξοδο του header.
4.3.2.5
Βίδωμα του συγκρατητή του στοπ (αν υπάρχει).
4.4.1
Τοποθέτηση δεύτερου header κατά τη διεύθ. ΙΙ.
Πλάνο 4.4.1
για full-size ή φινάκια ή οριζόντιο: 4.4.1.1
για μαίανδρο:
4.4.1.2 > 4.4.1.3 > 4.4.1.4 > 4.4.1.5
4.4.1.1
Τοποθέτηση κάθε συγκρατητή.
Πλάνο 4.4.1.1
4.3.1.1.1 >
4.3.1.1.2 >
4.3.1.1.3 >
4.3.1.1.4
4.4.1.1.1
Ξεβίδωμα βίδας συγκρατητή.
4.4.1.1.2
Χρήση κατάλληλου αριθμού ροδελών για ρύθμιση του ύψους του συγκρατητή (αν χρειαστεί, πχ. σε περίπτωση διαμέτρου διαφορετικής από ⌀22).
4.4.1.1.3
Μεταφορά του συγκρατητή σε κατάλληλη θέση (ένας σε κάθε άκρο και αν χρειαστεί ένας ακόμα στο κέντρο).
4.4.1.1.4
Τοποθέτηση και βίδωμα βίδας συγκρατητή.
4.4.1.2
Τοποθέτηση των δύο ξύλινων πλαισίων.
4.4.1.3
Ταίριασμα των γωνιών του μαιάνδρου riser στις εσοχές του πλαισίου, και τελικά τοποθέτηση του πάνω header στα πάνω άκρα των πλαισίων.
4.4.1.4
Τοποθέτηση των μεταλλικών μπαρών πάνω από τα ξύλινα πλαίσια.
4.4.1.5
2.4.2.4
Κόλληση με χαρτοταινία ενός στοπ στην τράπεζα, στην δεξιά πλευρά του φινακιού (σε θέση κοντά στην πάνω μπάρα).


Κόλληση ενός στοπ πάνω στην τράπεζα στο κάτω άκρο κάθε πλαισίου.
4.4.2.1 Μετακίνηση header κατά την διεύθ. Ι.
4.4.2.2 Μέτρηση, με χαρακάκι, απόστασης πρώτου riser από πλευρά φύλλου (να είναι ίση με αυτή κοντά στον άλλο header).
4.4.2.3 Μέτρηση, με χαρακάκι, του μήκους μίας ανοιχτής εξόδου header.
4.4.2.4 Κόλληση ενός στοπ πάνω στην τράπεζα, σε ανοιχτή έξοδο του header.
4.4.2.5 Τοποθέτηση και βίδωμα του συγκρατητή του στοπ (αν υπάρχει).
4.4.2.6 Ο header είναι ήδη σωστά τοποθετημένος με τη χρήση των ξύλινων πλαισίων (όπως περιγράφηκε στο προηγούμενο βήμα).
4.4.2.7 Μεταφορά της πάνω μπάρας κατά τη διεύθ. ΙΙ μέχρι η πλευρά της να ακουμπήσει το πάνω header.
4.4.2.8 Βίδωμα βιδών μπάρας.
4.4.2.9 Βίδωμα των συγκρατητών στα άκρα της πάνω μπάρας.
Πλάνο 4.4.2
για full-size ή φινάκια ή οριζόντιο:
4.4.1.1 > 4.4.2.2 > 4.4.2.3 > 4.4.2.4 > 4.4.2.5
για μαίανδρο:
4.4.1.6 > 4.4.1.7 > 4.4.1.8 > 4.4.1.9
4.4.2 Τοποθέτηση δεύτερου header κατά τη διεύθ. Ι (εξασφάλιση ευθυγράμμισης των riser).
4.4 Τοποθέτηση του δεύτερου header.
Πλάνο 4.4
4.4.1 > 4.4.2
5.1 Ξεβίδωμα του παξιμαδιού.
5.2 Αφαίρεση της βίδας, του παξιμαδιού, και της ρόδας.
5.3 Τοποθέτηση της άλλης ρόδας, με την ίδια βίδα και το παξιμάδι.
5.4 Βίδωμα του παξιμαδιού.
5.5 Για μετάβαση από ⌀8 σε ⌀15 χρειάζονται 5 αντιωρολογιακές περιστροφές του knob που βρίσκεται από πάνω. Από ⌀15 σε ⌀8 το αντίστροφο.
5.6 Δοκιμή και έλεγχος ομαλής περιστροφής της ρόδας.
5. Αλλαγή της ρόδας του laser (μόνο αν αλλάζει η διάμετρος των riser).
Πλάνο 5
5.1 > 5.2 > 5.3 > 5.4 > 5.5 > 5.6
6.1 Επιλογή κατάλληλου προγράμματος για το συγκεκριμένο (ή παρόμοιο προηγούμενο) προϊόν.
6.2 Εισαγωγή νέου βήματος (ή λοιπών χαρακτηριστικών αν χρειάζεται).
6.3 Καλιμπράρισμα του laser (σε περίπτωση που δεν προϋπάρχει πρόγραμμα για αυτό ακριβώς το προϊόν).
6. Προετοιμασία προγράμματος του laser.
Πλάνο 6
αρχικά: 6.1 >
εάν δεν υπάρχει κατάλληλο πρόγραμμα: > 6.2 > 6.3
6.3.1 Εισαγωγή, με δοκιμή και σφάλμα, αρχικού offset για το πρώτο riser.
6.3.2 Βηματική εκτέλεση μέχρι το κατέβασμα του laser πάνω στο πρώτο riser.
6.3.3 Έλεγχος (οπτικά) σωστής εφαρμογής.
6.3.4 Για το πρώτο και το δεύτερο riser (και πρόσω και πίσω διαδρομή).
Πλάνο 6.3
6.3.1 > 6.3.2 > 6.3.3 > 6.3.4
6.3.4.1 Βηματική εκτέλεση μέχρι τη συγκόλληση του πρώτου riser.
6.3.4.2 Οπτικός έλεγχος, με φακό, της ποιότητας της συγκόλλησης.
Πλάνο 6.3.4
επαναληπτικά τα: 6.3.4.1 > 6.3.4.2
7.1 Μέτρηση μήκους εξόδου header.
7.2 Μέτρηση απόστασης πρώτου riser από την άκρη του φύλλου.
7.3 Μέτρηση των διαγωνίων του grid ώστε να ελεγχθεί η καθετότητα των σωληνών.
7.4 Έλεγχος, οπτικός και με κατάλληλο εργαλείο, της συγκόλλησης.
7. Ποιοτικός έλεγχος.
Πλάνο 7
7.1 / 7.2 / 7.3 / 7.4
8.1 Τοποθέτηση κάθε εργαλείου που χρησιμοποιήθηκε πίσω στη θέση του.
8. Τακτοποίηση του εξοπλισμού.
Πλάνο 8
8.1

Τα βασικά καθήκοντα σε κάθε περίπτωση είναι η συγκέντρωση των εργαλείων, η εγκατάσταση των στηρίξεων του φύλλου (ή των φινακιών), η τοποθέτηση της πάνω μπάρας, στη συνέχεια η τοποθέτηση των στηρίξεων του πλέγματος σωληνών, σπανίως η αλλαγή ρόδας του laser και τελικά, χωρίς αυστηρή διαδοχή, η προετοιμασία του προγράμματος του laser και η τακτοποίηση των εργαλείων. Παρόλο που η διατύπωση της διαδικασίας με αυτά μόνο τα βήματα είναι σύντομη, ευκολομνημόνευτη, και επιτρέπει τη μέγιστη ευελιξία μεταξύ των απουσιάζουν αρκετές λεπτομέρειες που αποτελούν και τις ειδοποιούς διαφορές μεταξύ των διαφορετικών τύπων προϊόντων (λχ. η απαραίτητη τοποθέτηση των οδηγών της κάτω πλευράς στην περίπτωση των φινακιών (2.3) η οποία δεν χρειάζεται σε κάποιο άλλο προϊόν). Από το επόμενο κιόλας επίπεδο είναι εμφανής η διαφοροποίηση εξαιτίας του τύπου προϊόντος των υπο-καθηκόντων σε τρία από τα βασικά καθήκοντα (στα 1., 2., και 3.), ενώ υπάρχουν και προαιρετικά βήματα που εξαρτώνται από άλλες παραμέτρους (λχ. τα 6.2 και 6.3, σε περίπτωση που δεν υπάρχει έτοιμο το απαραίτητο πρόγραμμα για το laser).

Σε όσο πιο χαμηλά επίπεδα ανατρέχει κανείς κατά την εκτέλεση της διαδικασίας τόσο μειώνεται το κίνδυνος σφάλματος λόγω αμέλειας ή λανθασμένης σειράς εκτέλεσης (λχ. σφάλμα: τοποθέτηση του header πρώτα κατά την αξονική διεύθυνση, I, και ύστερα κατά την ακτινική, II), και αποφεύγεται ο επιπλέον χρόνος διόρθωσής του. Φυσικά θα ήταν χρονοβόρο να ελέγχεται ολόκληρο το διάγραμμα σε κάθε σετάρισμα παρά τη μείωση πιθανότητας σφάλματος. Συνεπώς παρόλο που δεν μπορεί να οριστεί μονοσήμαντα το βέλτιστο βάθος επιπέδου με το οποίο επιτυγχάνεται η μεγαλύτερη δυνατή ταχύτητα χάρη στην γρήγορη ανάκληση των βημάτων και ταυτόχρονα την συστηματικά σωστή εκτέλεσή τους, ένας καλός συμβιβασμός θα ήταν η χρήση των βημάτων του δεύτερου επιπέδου ως πρώτη αναφορά, και σε περίπτωση αβεβαιότητας μπορεί κανείς να ανατρέξει ύστερα στα βήματα των χαμηλότερων επιπέδων.

Αξίζει να σημειωθεί ότι υπάρχουν ποικίλα κομβικά σημεία στις απαραίτητες ενέργειες κάθε σταδίου τα οποία εξαρτώνται συχνά από ανεξάρτητες, μεταξύ τους, παραμέτρους (λχ. οριζόντιο προϊόν ταυτόχρονα με riser ⌀15). Ένας έμπειρος χειριστής είναι σε θέση να ανακαλέσει όλα τα κατάλληλα βήματα για οποιαδήποτε περίπτωση προϊόντος και να τα υλοποιήσει με μικρή πιθανότητα σφάλματος. Σε περίπτωση όμως που απαιτείται μεγαλύτερη ευελιξία χρήσης της εγκατάστασης και από το υπόλοιπο μη έμπειρο προσωπικό αλλά και συντονισμένης εργασίας δύο χειριστών κατά το σετάρισμα κρίνεται αναγκαία η τυποποίηση των διαδικασιών.

### 3.3.2 Διερεύνηση βελτίωσης εντολής εργασίας

Με στόχο την μείωση του χρόνου σεταρίσματος της μηχανής laser για κάθε περίπτωση προϊόντος επιλέχθηκε ως μία λύση η βελτίωση της μορφής των πληροφοριών που δίνονται στους χειριστές μέσω της εντολής εργασίας. Η μορφή της υπάρχουσας εντολής εργασίας είναι ένας πίνακας στον οποίον μεταξύ άλλων πληροφοριών περιέχονται και οι καθοριστικές διαστάσεις του προϊόντος. Εξετάσθηκε το ενδεχόμενο να παρέχονται οι διαστάσεις του προϊόντος σε σχηματική μορφή σε συνδυασμό με την απαραίτητη διαδοχή κάποιον ή όλων των απαραίτητων ενεργειών για την ολοκλήρωση του σεταρίσματος. Η πλήρης διαδικασία σεταρίσματος έχει αναλυθεί σε προηγούμενη αναφορά.

Οι περιπτώσεις χρηστών, οι οποίοι θα ωφεληθούν τελικά από αυτή τη διαδικασία, ποικίλουν τόσο λόγω επαγγελματικής εμπειρίας όσο και ατομικών προτιμήσεων. Για αυτόν τον λόγο κρίθηκε απαραίτητη η επικοινωνία με το προσωπικό κατά τη διαμόρφωση της βέλτιστης εντολής εργασίας. Πραγματοποιήθηκαν συνεντεύξεις με τους τρεις (συχνότερους) χειριστές της μηχανής laser καθώς και με τον υπεύθυνο της εν λόγω μονάδας παραγωγής ώστε να καταγραφούν όλες οι ανάγκες οι οποίες θα πρέπει να καλυφθούν. Στη συνέχεια αυτές χρησιμοποιήθηκαν συγκεκριμένα στην επιλογή των ενδεικνυόμενων διαστάσεων πάνω στο ενδεικτικό σχέδιο και στην επιλογή του βέλτιστου, κατά περίπτωση, αφαιρετικού επιπέδου των ενδεικνυόμενων βημάτων, εάν χρειάζονται.

Ερωτήθηκαν τρεις χειριστές της μηχανής laser και ο υπεύθυνος της μονάδας παραγωγής για τις απαραίτητες ενδείξεις που θεωρούν χρήσιμες σε κάθε αλλαγή προϊόντος τόσο σχετικά με τα χαρακτηριστικά του προϊόντος όσο και με την διαδικασία που πρέπει να ακολουθηθεί. Ο πρώτος χειριστής (χειριστής 1) είναι και ο κύριος χειριστής καθώς συχνότερα αυτός μοιράζει τις εργασίες σε κάποιο σετάρισμα. Ο δεύτερος χειριστής (χειριστής 2) απασχολούνταν παλαιότερα αποκλειστικά με την μηχανή laser αλλά πλέον απασχολείται σε διάφορες θέσεις και περιστασιακά και σε αυτή. Ο τρίτος χειριστής (χειριστής 3) λαμβάνει βοηθητικό ρόλο στον κύριο χειριστή και απασχολείται αποκλειστικά με την μηχανή laser.

Και οι τρεις χειριστές δήλωσαν ότι τους είναι χρήσιμο να φαίνονται οι απαραίτητες ονομαστικές διαστάσεις σχηματικά. Επιπλέον δήλωσαν ότι είναι χρήσιμο να αναγράφονται οι ονομαστικές διαστάσεις σε όλα τα σημεία στο οποία θα πρέπει να μετρηθούν κατά το σετάρισμα με αριθμημένη σειρά που αντιστοιχεί στα αντίστοιχα βήματα. Βέβαια συχνά λόγω ανακρίβειας (υπό του χιλιοστού) δεν μπορούν να χρησιμοποιηθούν με αυστηρότητα κάποιες ονομαστικές διαστάσεις, όπως λ.χ. το μήκος του φύλλου πέρα από τον header που είναι της τάξης των $0 - 5\,\mathrm{mm}$ (αναλόγως το προϊόν) και συμμετρικό στις δύο πλευρές, το οποίο ισομοιράζεται τη στιγμή του σεταρίσματος σε περίπτωση που το συνολικό μήκος του φύλλου δεν είναι ακριβώς ίσο με το ονομαστικό. Ακόμη, απαραίτητες κρίθηκαν και οι διαστάσεις του φύλλου, οι οποίες χρησιμοποιούνται έμμεσα για την τοποθέτηση των στοπ (λ.χ. σε περίπτωση φινακιών) ή για την μέτρηση λ.χ. του προαναφερθέντος μήκους μέσω της διαφοράς των αποστάσεων της άκρης του φύλλου και της πλευράς του header από την εσοχή του προφίλ αλουμινίου στην πλευρά της τράπεζας.

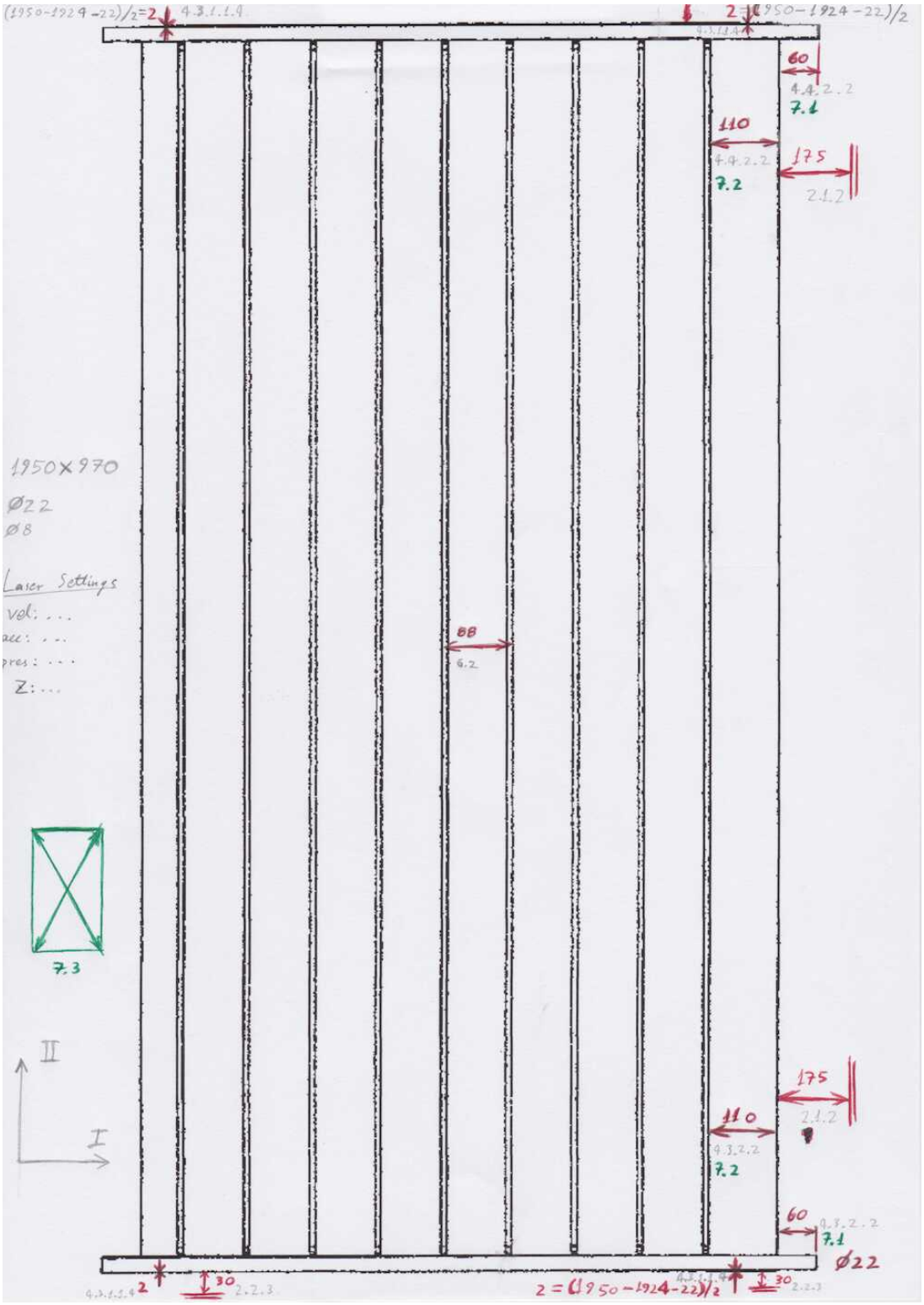
Επιπλέον οι χειριστές 2 και 3 εκδήλωσαν την επιθυμία εκτός από τις ονομαστικές διαστάσεις, όπως περιγράφηκαν νωρίτερα, να αναγράφονται και οι αποστάσεις των πλευρών του φύλλου από τις άκρες της τράπεζας. Η παρουσία τους και μόνο λειτουργεί ως υπενθύμιση για την σωστή τοποθέτηση των στοπ των φύλλων ακόμα και αν αυτά τυχαίνει να είναι ήδη τοποθετημένα. Συνήθως χρειάζεται μετακίνησή τους μόνο για οριζόντια ή μεγάλου πλάτους προϊόντα και οι θέσεις είναι προκαθορισμένες. Επιπλέον ζητήθηκε να αναγράφονται οι προκαθορισμένες για κάθε συνδυασμό φύλλου και πλέγματος συνθήκες λειτουργίας του laser

(ταχύτητα, επιτάχυνση, πίεση, και "Z"). Παρόλο που αυτές δεν είναι σχεδιαστικές παράμετροι είναι προκαθορισμένες (πλέον) για κάθε παραλλαγή των τεμαχίων (υλικό και πάχος φύλλου, διάμετρος των riser) και επειδή δεν χρειάζεται αλλαγή τους σε κάθε προϊόν στις ορισμένες περιπτώσεις που χρειάζεται γίνεται αναδρομή σε σημειώσεις από παλαιότερα προϊόντα, ή καλείται ο χειριστής 1 ο οποίος είναι ο μόνος που τις θυμάται.

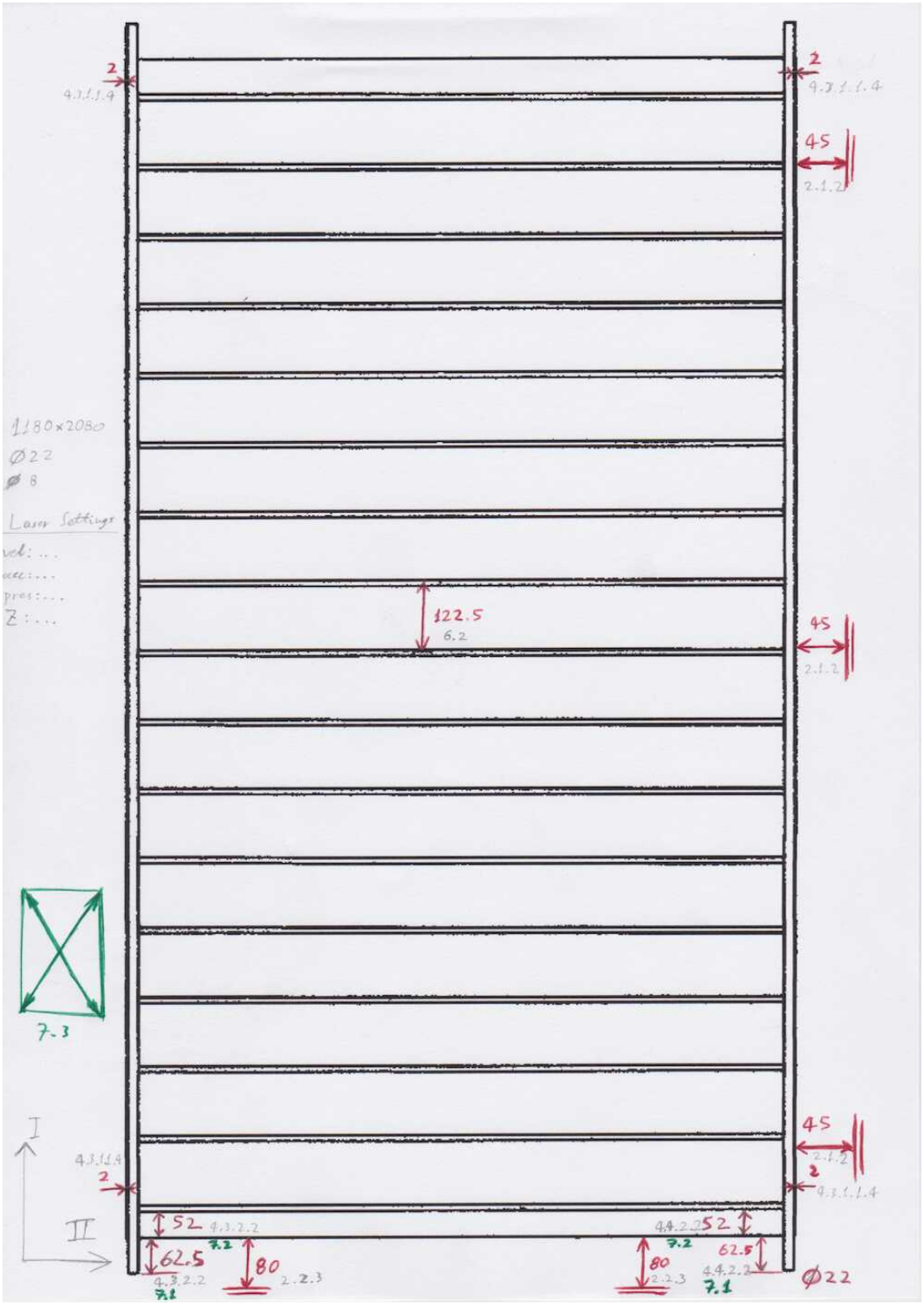
Όσον αφορά την ένδειξη απαραίτητων βημάτων οι χειριστές 1 και 2 την θεώρησαν περιττή σε κάθε επίπεδο. Δεν θα εμπόδιζε την διαδικασία, απλώς δεν θα αφιέρωναν καθόλου χρόνο στην ανάγνωσή της (δεδομένου ότι θα δίνεται και το σχέδιο όπως αναλύθηκε παραπάνω). Ο χειριστής 3 θεώρησε χρήσιμη την ένδειξη μόνο των καθηκόντων ανώτερων επιπέδων με προαιρετική επιλογή ανάγνωσης συγκεκριμένων υπο-καθηκόντων. Καθώς είχε χρειασθεί να προετοιμάσει την μηχανή λίγες φορές στο παρελθόν και γνωρίζει μερικώς την διαδικασία δεν θεωρεί απαραίτητη την ένδειξη όλων των βασικών βημάτων. Και οι τρεις χειριστές έκριναν ότι για κάποιον αρχάριο (στον οποίο θα χρειαζόταν να υποδείξουν πρώτη φορά τη διαδικασία) θα χρησίμευε η ένδειξη της συνολικής διαδικασίας σεταρίσματος.

Σε συμφωνία με τα παραπάνω ο υπεύθυνος παραγωγής (ο οποίος απασχολούνταν στην μηχανή laser παλαιότερα) πρόσθεσε ότι είναι σημαντική η ένδειξη (υπενθύμιση) του ποιοτικού ελέγχου. Μετά την παραγωγή των πρώτων δύο τεμαχίων κάθε νέας παλέτας πραγματοποιείται έλεγχος όλων των προηγουμένως μετρημένων διαστάσεων (εκτός από αυτές τις τοποθέτησης του φύλλου) και κυρίως η μέτρηση των διαγωνίων, για εξασφάλιση της καθετότητας του πλέγματος. Συνεπώς, στα σχήματα 8 και 9 φαίνονται δύο παραδείγματα σχεδίων με τις απαραίτητες ενδείξεις. Δεν εκδηλώθηκε ιδιαίτερη προτίμηση σχετικά με το μέσο στο οποίο θα αποτυπώνονταν τα σχέδια.

Θεωρήθηκε ότι τα σχέδια θα συνοδεύονται και από έντυπη ή ηλεκτρονική ένδειξη των βημάτων εκκινώντας από το ανώτερο επίπεδο ώστε αν χρειαστεί να μπορούν να αναζητηθούν τα σχετικά αριθμημένα βήματα κάθε μέτρησης όπως φαίνεται στα σχήματα. Στους κωδικούς των σχετικών βημάτων, μεταξύ άλλων, βρίσκονται και οι κωδικοί των βημάτων ποιοτικού ελέγχου ως υπενθύμιση.



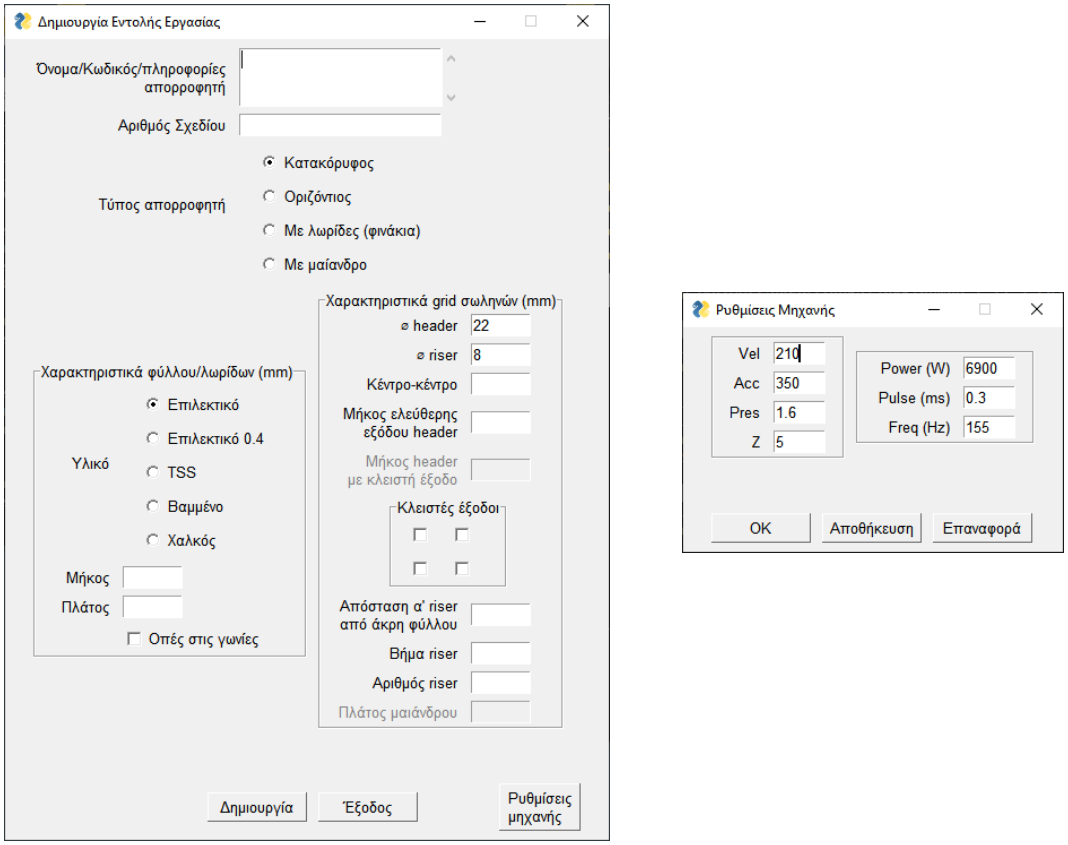

Σχήμα 8: Σχηματική αναπαράσταση προϊόντος full-size με τις ονομαστικές διαστάσεις, με κόκκινο, και τους αριθμούς των σχετικών βημάτων, με γκρι και πράσινο (τα βήματα ποιοτικού ελέγχου).




Σχήμα 9: Σχηματική αναπαράσταση οριζόντιου προϊόντος με τις ονομαστικές διαστάσεις, με κόκκινο, και τους αριθμούς των σχετικών βημάτων, με γκρι και πράσινο (τα βήματα ποιοτικού ελέγχου).

### 3.3.3 Κατασκευή προγράμματος παραγωγής εντολής εργασίας και οδηγιών προετοιμασίας

Για την αυτόματη παραγωγή ενός παραμετρικού σχεδίου το οποίο θα συμπεριληφθεί σε μία βελτιωμένη εντολή εργασίας κατασκευάστηκε ένα πρόγραμμα στο οποίο εισάγονται οι απαραίτητες πληροφορίες και παράγεται αυτόματα ένα σχέδιο του προϊόντος το οποίο συνοδεύεται από πληροφορίες σχετικά με τη ρύθμιση της μηχανής και του laser. Το πρόγραμμα γράφτηκε σε Python 3.7 με χρήση του πακέτου PySimpleGUI για την δημιουργία του interface και του πακέτου Pillow (των αντικειμένων του Image και ImageDraw) για την δημιουργία του σχεδίου και μεταγλωττίστηκε με τον PyInstaller σε εκτελέσιμο για MS Windows 7 32bit και MS Windows 10 64bit. Ο κώδικας βρίσκεται στο παράρτημα Α και η διεπιφάνεια στην οποία εισάγονται οι πληροφορίες φαίνεται στην εικόνα 10.




Εικόνα 10: Κεντρικό παράθυρο και παράθυρο με τις ρυθμίσεις της μηχανής laser από το πρόγραμμα παραγωγής εντολής εργασίας.

Ως είσοδος δίνονται από το χρήστη όλα τα απαραίτητα γεωμετρικά χαρακτηριστικά, καθώς επίσης και το υλικό και οι επιθυμητές συνθήκες λειτουργίες του laser. Αξιοποιήθηκε η χρησιμοποιούμενη τυποποίηση στην ονομασία των προϊόντων ώστε να επιλέγεται αυτόματα το υλικό του φύλλου ή η περίπτωση προϊόντος με λωρίδες. Συγκεκριμένα μόλις στο όνομα του συλλέκτη (τουλάχιστον στην πρώτη γραμμή του πεδίου, αν υπάρχουν πολλαπλές) εμφανιστεί η κατάληξη «FPNXXX», όπου «XXX» τρεις οποιοιδήποτε αριθμοί και «N» ίσο με 2, 3, 4, ή 8 τότε επιλέγονται οι αντίστοιχες ρυθμίσεις σύμφωνα με τον πίνακα 1.

Πίνακας 1: Αντιστοίχιση κωδικοποίησης ονόματος με ρύθμιση μηχανής.

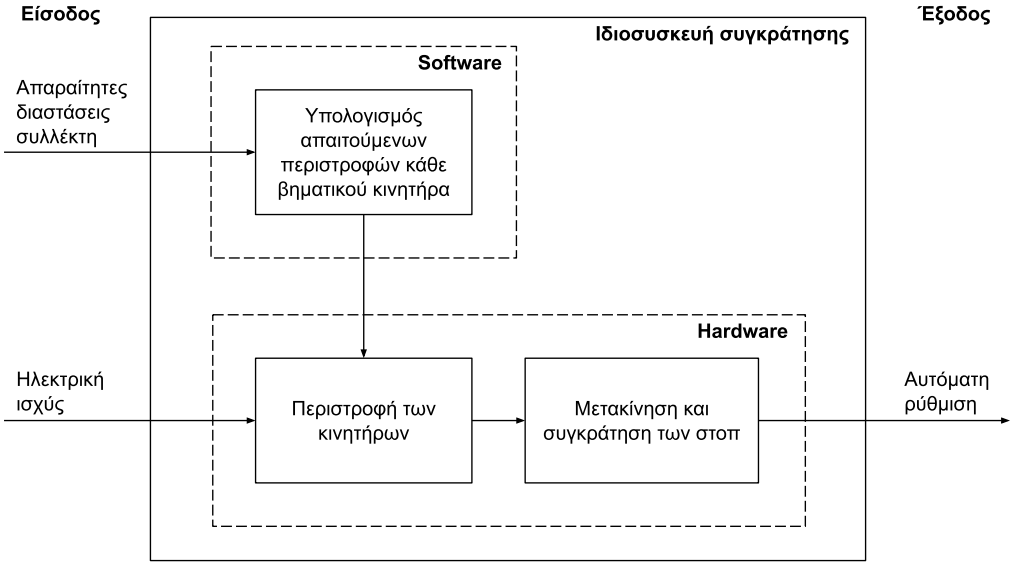
| N | Ρυθμίσεις |
|---|---|
| 2 | Υλικό: Επιλεκτικό |
| 3 | Υλικό: TSS |
| 4 | Υλικό: Βαμμένο |
| | Τύπος: Με |
| 8 | λωρίδες, Υλικό: |
| | Επιλεκτικό |

Για παράδειγμα εισάγοντας ένα όνομα με κατάληξη
ενδοτςΦΠ8011 θα επιλεχθεί αυτόματα ο τύπος συλλέκτη με λωρίδες και με επιλεκτικό φύλλο. Αυτές οι ρυθμίσεις μπορούν να αλλαχθούν ύστερα και με το χέρι. Οι υπόλοιπες επιλογές που πρέπει υποχρεωτικά να συμπληρωθούν είναι το μήκος και το πλάτος του φύλλου, η απόσταση κέντρο-κέντρο των header, το μήκος των ελεύθερων εξόδων των header, η απόσταση του α΄ riser από την άκρη του φύλλου, και το βήμα και το πλήθος των riser. Επιλέγοντας κάποια κλειστή είσοδο ελευθερώνεται το κελί για το μήκος του header με την κλειστή είσοδο ενώ επιλέγοντας προϊόν με μαίανδρο ελευθερώνεται το κελί για το πλάτος του μαιάνδρου. Ομοίως, αν επιλεχθεί προϊόν με λωρίδες τότε απενεργοποιούνται οι επιλογές «TSS», «Βαμμένο», και «Χαλκός» για το υλικό, και η επιλογή οπών στις γωνίες του φύλλου. Μόλις συμπληρωθούν οι τιμές και επιλεγεί «Δημιουργία» τότε πραγματοποιούνται κάποιοι βασικοί έλεγχοι των τιμών και εμφανίζονται τα κατάλληλα μηνύματα σφάλματος πάνω από το κουμπί ώστε να εξασφαλιστεί ότι δεν έχει γίνει κάποιο λάθος και ότι το παραγόμενο σχέδιο αντιστοιχεί σε φυσικά υλοποιήσιμο συλλέκτη.

Επιπλέον υπάρχει η δυνατότητα Διευκρίνησης των επιθυμητών ρυθμίσεων λειτουργίας της μηχανής laser. Επιλέγοντας «Ρυθμίσεις μηχανής» εμφανίζεται ένα νέο παράθυρο με τις συνήθεις τιμές των συνθηκών ανάλογα με το υλικό του φύλλου (οι Power, Pulse, Freq) ανάλογα με τη διάμετρο των riser (το Z), οι οποίες μπορούν να τροποποιηθούν. Οι υπάρχουσες τροποποιημένες τιμές μπορούν να αποθηκευτούν ως νέες συνήθειες σε εξωτερικό αρχείο κειμένου (που πρέπει να βρίσκεται στην ίδια τοποθεσία με το εκτελέσιμο) και να φορτωθούν αυτόματα σε κάθε εκκίνηση του προγράμματος (σε περίπτωση απουσίας του αρχείου απλά χρησιμοποιούνται οι αρχικά καθορισμένες συνήθεις τιμές). Μόλις επιλεγεί «Δημιουργία» και επιλεγεί η τοποθεσία αποθήκευσης παράγεται ένα αρχείο pdf με τρεις σελίδες. Στο παραγόμενο σχέδιο στην πρώτη σελίδα αναγράφονται με μπλε οι απαραίτητες διαστάσεις για τη ρύθμιση της μηχανής ακολουθούμενες από τα κατάλληλα βήματα στα οποία χρησιμοποιούνται. Επιπλέον φαίνονται με μαύρο κάποιες βασικές διαστάσεις οι οποίες είναι έμμεσα απαραίτητες για την ανίχνευση λαθών σε κάποια προηγούμενη φάση ή για επιπλέον επαλήθευση. Κάτω αριστερά αναγράφεται το σύστημα συντεταγμένων που χρησιμοποιείται στις οδηγίες ρύθμισης (προσαρμοσμένο ανάλογα με τη διεύθυνση των header). Κάτω δεξιά φαίνονται οι προτεινόμενες συνθήκες λειτουργίας του laser. Στην δεύτερη σελίδα αναγράφονται τα κατά περίπτωση απαιτούμενα βήματα τα οποία καλό είναι να έχει υπόψη του ο συντονιστής της διαδικασίας του σεταρίσματος (η διαδοχή εδώ είναι αυστηρή). Στην τρίτη σελίδα αναγράφονται τα απαραίτητα υποκαθήκοντα που πρέπει να πραγματοποιηθούν από τους υπόλοιπους χειριστές και η διευκρινίζεται η ελευθερία εναλλαγής της διαδοχής τους όπου υπάρχει. Για επιπλέον εμβάθυνση σε κάποιο βήμα οι χειριστές μπορούν να ανατρέξουν στο αρχείο «οδηγίες.pdf» στο οποίο βρίσκονται τα βήματα με υπερσυνδέσμους και συνεπώς μπορεί να αναζητηθεί οποιοδήποτε βήμα εφόσον είναι γνωστός ο κωδικός του.

## 3.4 Προσδιορισμός τεχνικών προδιαγραφών και περιορισμών

Πριν πραγματοποιηθεί ο σχεδιασμός κάποιας πιθανής λύσης προσδιορίστηκαν οι προδιαγραφές τις οποίες πρέπει να ικανοποιεί η ιδιοσυσκευή, ώστε να καθίσταται χρηστική, και οι περιορισμοί στους οποίους υπόκειται, ώστε να μπορεί να ενσωματωθεί και να λειτουργήσει πάνω στις τράπεζες της υπάρχουσας μηχανής laser. Σε αυτό το σημείο χρειάζεται να διατυπωθεί το δομικό διάγραμμα λειτουργίας της ιδιοσυσκευής στο οποίο είναι εμφανή τα επιμέρους υποσυστήματα και οι λειτουργίες που εκτελούν. Όπως φαίνεται στο σχήμα 10 τα δύο κύρια υποσυστήματα είναι αυτό του λογισμικού και αυτό της φυσικής διάταξης. Ο ρόλος του λογισμικού είναι η μετατροπή της πληροφορίας των απαραίτητων διαστάσεων ενός συλλέκτη, όπως τις γνωρίζουν οι χειριστές, σε απαιτούμενες στροφές κινητήρων και συνεπώς σε κινήσεις των μελών της φυσικής διάταξης. Ο βασικός ρόλος της διάταξης είναι η δυνατότητα κατάλληλης κίνησης των μελών της (σε αποδεκτό χρονικό διάστημα) και η συγκράτηση της θέσης τους ώστε τελικά να πραγματοποιηθεί ομαλά η συγκόλληση και μεταφορά του συλλέκτη.




Σχήμα 10: Δομικό διάγραμμα λειτουργίας της ιδιοσυσκευής συγκράτησης.

Συνεπώς, με στόχο την κάλυψη αυτού ρόλου της διάταξης ορίστηκαν οι απαιτήσεις και οι περιορισμοί που πρέπει να πληροί. Οι τύποι προϊόντων στους οποίους πρέπει να χρησιμοποιηθεί η διάταξη είναι αυτοί του κατακόρυφου και του οριζόντιου συλλέκτη. Παράλληλα είναι επιθυμητό να μην εμποδίζεται το σετάρισμα σε δεύτερο χρόνο, ακόμα και χειροκίνητα (με επιπλέον ιδιοσυσκευές), για συλλέκτη με μαίανδρο.

Οι απαιτήσεις, διαμορφωμένες με βάση την υπάρχουσα διαδικασία σεταρίσματος (καθώς αυτή πρόκειται να αντικατασταθεί), διατυπώθηκαν ως εξής:

i Περιορισμός της επίπεδης κίνησης του φύλλου πάνω στην τράπεζα (σε δύο διευθύνσεις).

ii Για κάθε header: τοποθέτηση σε κατάλληλη θέση σε σχέση με το φύλλο και περιορισμός της επίπεδης κίνησής (ώστε να εμποδίζεται η μετακίνηση του πλέγματος).

iii Για κάθε header: επαρκής συγκράτηση κατά την κατακόρυφη στην τράπεζα διεύθυνση (ώστε να μην υπάρχει παραμόρφωση κατά τη συγκόλληση κάθε riser εξαιτίας παραμενουσών τάσεων)

iv Δυνατότητα τοποθέτησης συλλεκτών με ελάχιστη απόσταση κέντρο-κέντρο ίση με 1200 mm και ελάχιστο μήκος header ίσο με 900 mm.

Οι περιορισμοί τέθηκαν ως εξής:

i Να μην υπάρχει σύγκρουση της ιδιοσυσκευής με τα υπόλοιπα μέλη της μηχανής laser, κατά την κίνηση των τραπεζών.

ii Να μην υπάρχει σύγκρουση με τις κεφαλές laser κατά τη συγκόλληση.

iii Να μπορεί να στερεωθεί η ιδιοσυσκευή πάνω στις υπάρχουσες τράπεζες.

iv Να επιτρέπει το σετάρισμα και το φόρτωμα συλλέκτη με μαίανδρο.

# 4 Διαδικασία σχεδιασμού

## 4.1 Επιλογή concept

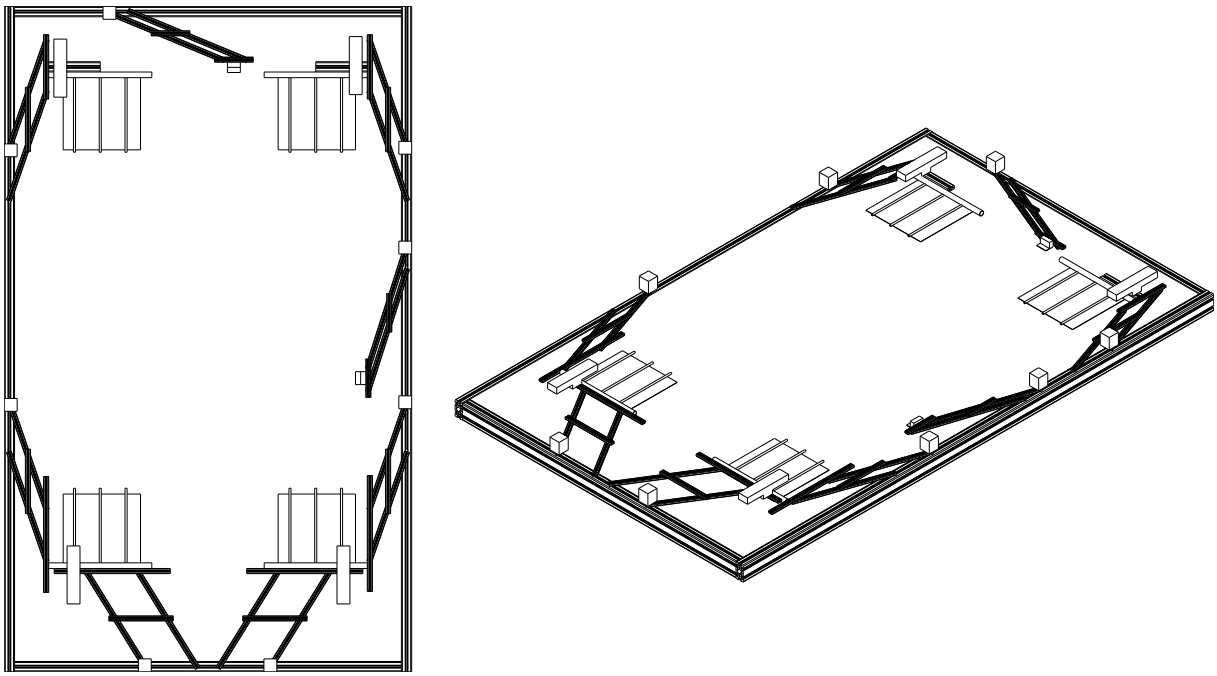
Για την δημιουργία ενός ικανοποιητικού σχεδιασμού της ιδιοσυσκευής συγκράτησης σε σχεδιαστικό πρόγραμμα, επιχειρήθηκε αρχικά η ανεύρεση λύσεων για κάθε μία από τις απαιτήσεις που διατυπώθηκαν και ταυτόχρονα ελέγχθηκε η ικανοποίηση κάθε περιορισμού. Όσον αφορά την απαίτηση (i) για την επαρκή στήριξη του φύλλου χρειάζεται η μία οριζόντια και η μία κάθετη πλευρά του να εφάπτονται σε μία γεωμετρία κατά παρόμοιο τρόπο με τα υπάρχοντα «στοπ» που χρησιμοποιούνται. Για την απαίτηση (ii), και πάλι με βάση την υπάρχουσα μέθοδο σεταρίσματος, θα πρέπει σε μία τουλάχιστον ελεύθερη έξοδο κάθε header να βρίσκονται δύο «στοπ», ένα για κάθε διεύθυνση. Τοποθετώντας το πλέγμα ώστε αρχικά να εφάπτεται σε αυτά τα «στοπ», έχει εξασφαλιστεί η σωστή τοποθέτηση του πλέγματος σε σχέση με το φύλλο. Η απαίτηση (iii) θα πρέπει να καλυφθεί με χρήση συσκευών οι οποίες, στερεωμένες με κάποιον τρόπο στην τράπεζα, θα μπορούν να ασκήσουν επαρκή αντίσταση στην κατακόρυφη κίνηση των header κατά τη συγκόλληση. Δύο πιθανές λύσεις είναι η χρήση συγκρατητών τύπου quick clamp και η χρήση περιστρεφόμενων ελασμάτων παρόμοια με αυτά που χρησιμοποιούνται τώρα. Και οι δύο τύποι συγκρατητών, αν και μη αυτόματοι, μπορούν να ρυθμιστούν ώστε να πιέζουν επαρκώς τους header χωρίς να τους παραμορφώνουν. Η ικανοποίηση της απαίτησης (iv) εξαρτάται άμεσα από την ικανοποίηση των περιορισμών (i), (ii), και (iii), ανάλογα με την διάταξη των υπόλοιπων μελών της ιδιοσυσκευής.

Για την ικανοποίηση του περιορισμού (i) αρχικά προσδιορίστηκε ο μέγιστος χώρος μέσα στον οποίο πρέπει να βρίσκονται όλα τα μέλη της ιδιοσυσκευής. Καθώς η τράπεζα κινείται από και προς την θέση φόρτωσης πλαισιώνεται περιφερειακά από προφίλ αλουμινίου επομένως δεν πρέπει να προεξέχει κάποιο μέλος πέρα από την άκρη της τράπεζας. Το μέγιστο επιτρεπόμενο ύψος καθορίζεται, αρχικά, από την κίνηση των τραπεζών και συγκεκριμένα τη στιγμή που βρίσκονται η μία πάνω από την άλλη κατά την εναλλαγή τους, και μετρήθηκε ίσο με 87 mm. Επομένως ο επιτρεπτός χώρος ορίζεται αρχικά ως ένα παραλληλεπίπεδο πάνω από την τράπεζα, με πλάτος και μήκος ίσο με αυτής (1580 mm × 2580 mm), και ύψος ίσο με 80 mm (για επιπλέον ασφάλεια). Αυτός ο χώρος τροποποιείται περαιτέρω αν ληφθεί υπόψη ο περιορισμός (ii). Για ένα δεδομένο προϊόν θα πρέπει να λαμβάνεται υπόψη ο χώρος τον οποίο σαρώνει ο όγκος των κεφαλών laser ώστε να μην υπάρχει σύγκρουση με αυτές. Κυρίως ενδιαφέρουν μόνο οι ακραίες περιπτώσεις στις οποίες το laser κινείται πάνω σε κάθε ακραίο riser, τόσο σε κατακόρυφο όσο και σε οριζόντιο συλλέκτη. Ο έλεγχος αυτού του περιορισμού έγινε με κινηματική μελέτη στο σχεδιαστικό περιβάλλον εισάγοντας μία γεωμετρία με τις μέγιστες διαστάσεις που καταλαμβάνουν οι κεφαλές του laser. Για την ικανοποίηση του περιορισμού (iii) αρκεί κάθε μέλος της ιδιοσυσκευής να στερεώνεται επαρκώς βιδωμένο με T-slot nuts στις εσοχές των προφίλ αλουμινίου στις τέσσερις πλευρές της τράπεζας. Ο περιορισμός (iv) απαιτεί την ύπαρξη χώρου ώστε να μπορούν να τοποθετηθούν τα δύο ξύλινα πλαίσια συγκράτησης του μαιάνδρου riser.

Με βάση τις παραπάνω επιλογές είναι σαφή τα σημεία επαφής των τεμαχίων και της ιδιοσυσκευής (τα «στοπ» και οι συγκρατητές) ενώ δεν έχει προσδιοριστεί ο τρόπος αυτόματης τοποθέτησής τους. Συνεπώς επόμενο βήμα είναι ο προσδιορισμός της διάταξης με την οποία τα «στοπ» και οι συγκρατητές μετακινούνται και συγκρατούνται στις κατάλληλες θέσεις. Αυτή η διάταξη θα πρέπει τελικά να πληροί τους τέσσερις περιορισμούς και από την μορφή της θα εξαρτηθεί άμεσα η ικανοποίηση της απαίτησης (iv). Για τη μεταφορά των στοιχείων επιλέχθηκε να χρησιμοποιηθούν είτε (υβριδικοί) βηματικοί κινητήρες είτε σερβοκινητήρες καθώς και οι δύο τύποι παρέχουν επαρκή ακρίβεια κίνησης ενώ σε συνδυασμό με κιβώτιο

μετάδοσης ή lead screw μπορούν να αντιστέκονται σε μετακίνηση από εξωτερικές δυνάμεις. Παρήχθησαν στη συνέχεια τρεις σχεδιαστικές λύσεις και από εξέταση και σύγκρισή τους επιλέχθηκε η καταλληλότερη.

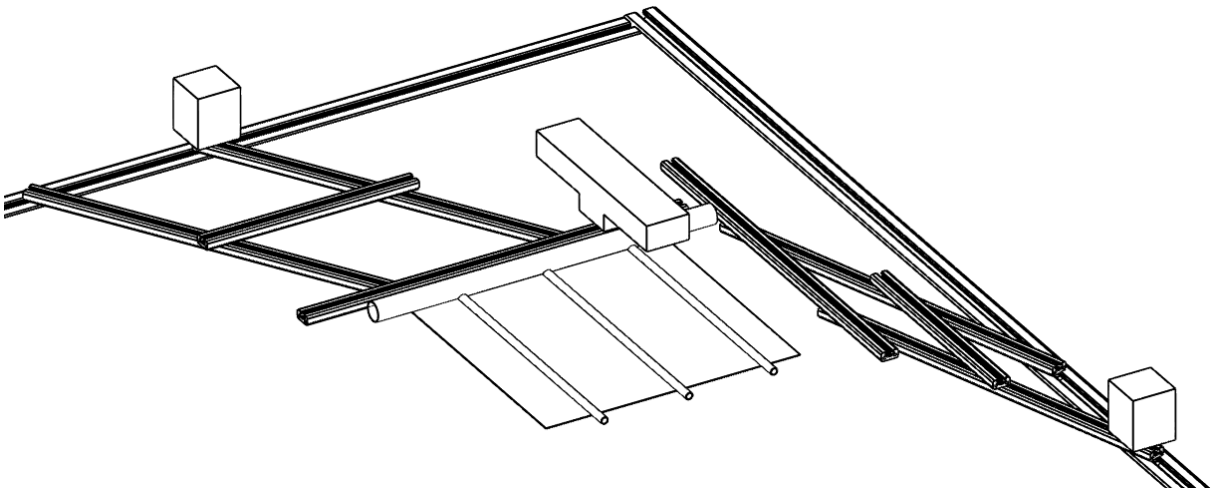
Η πρώτη λύση φαίνεται στο σχήμα 11. Στα σχήματα που ακολουθούν απεικονίζονται μόνο οι τέσσερις γωνίες του πλέγματος και του φύλλου ενός υποτιθέμενου συλλέκτη με μετακίνησή των οποίων μεταβάλλονται οι διαστάσεις του. Σε αυτή τη λύση η μετακίνηση των «στοπ» και των συγκρατητών quick clamps γίνεται μέσω περιστροφικής κίνησης. Τα περιστροφικά στοιχεία αποτελούν μία λύση οικονομική σε ύψος με δυνατότητα μεγάλου εύρους κίνησης. Οι βηματικοί κινητήρες, τοποθετημένοι κατακόρυφα, περιστρέφουν ζεύγη προφίλ αλουμινίου στις απολήξεις των οποίων βρίσκονται στερεωμένα τα «στοπ» και οι συγκρατητές. Οι header ακουμπούν πλευρικά σε προφίλ αλουμινίου ενώ το φύλλο ακουμπά σε δύο πλευρές σε «στοπ». Οι διαστάσεις και τα μήκη των προφίλ και των «στοπ» είναι ενδεικτικές, και τροποποιώντας τα μήκη τους μεταβάλλεται το εύρος διαστάσεων των συλλεκτών που μπορούν να φορτωθούν (λ.χ. για προφίλ με μικρά μήκη μπορούν να φορτωθούν κυρίως μεγάλοι συλλέκτες με μικρό εύρος μεταβολής διαστάσεων).



Σχήμα 11: Πρώτη σχεδιαστική λύση διάταξης ιδιοσυσκευής συγκράτησης. Απλοποιημένα απεικονίζονται με μπλε χρώμα οι βηματικοί κινητήρες, με ροζ οι συγκρατητές τύπου quick clamp, με κίτρινο τα «στοπ» των πλευρών του φύλλου, και με γκρι τα προφίλ αλουμινίου.

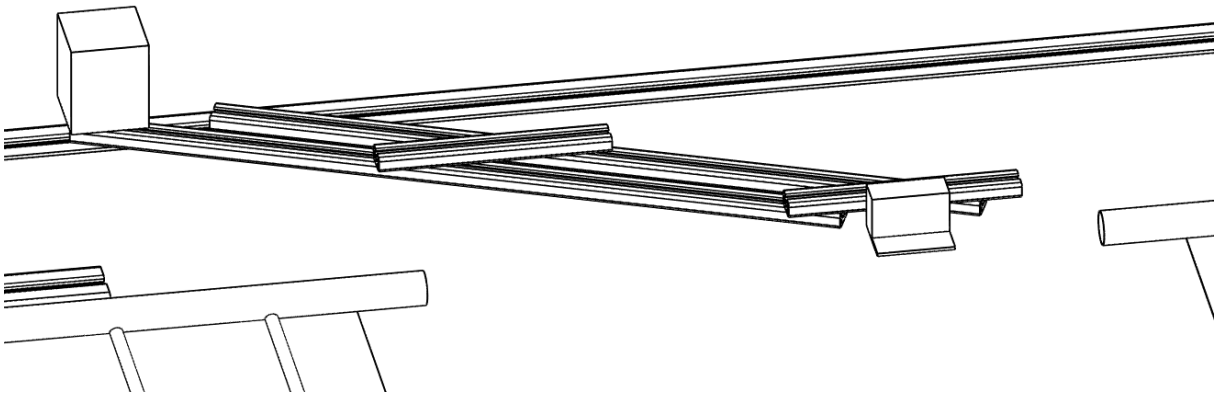
Προκειμένου να περιορίζεται η κίνηση του πλέγματος σωλήνων και στις δύο επίπεδες διευθύνσεις είναι απαραίτητο να υπάρχει επαφή με δύο στιβαρές επιφάνειες «στοπ». Επιπλέον οι δύο επιφάνειες χρειάζεται να τοποθετούνται με ακρίβεια στις κατάλληλες θέσεις χωρίς να εμποδίζουν στη φόρτωση τους χειριστές. Στην προκειμένη λύση η παράλληλη στις πλευρές της τράπεζας μετατόπιση αυτών των επιφανειών επιτυγχάνεται μέσω περιστροφικής κίνησης ζεύγους προφίλ αλουμινίου η οποία δίνεται από έναν (τουλάχιστον) κινητήρα πάνω στο προφίλ της πλευράς της τράπεζας. Ένα αποδεκτό εύρος διαστάσεων του πλέγματος μπορεί να επιτευχθεί με κατάλληλη επιλογή των μηκών των περιστρεφόμενων και κάθετων προφίλ ώστε να υπάρχει ομαλή συνεργασία των ζευγών «στοπ» σε κάθε γωνία. Εφόσον εξασφαλιστεί

η στιβαρότητα με την κατάλληλη επιλογή συνδετικών στοιχείων των προφίλ, η κατακόρυφη συγκράτηση του πλέγματος επιτυγχάνεται με χρήση quick clamp σε κάθε γωνία.



Σχήμα 12: Σύστημα τοποθέτησης των header της πρώτης σχεδιαστικής λύσης.

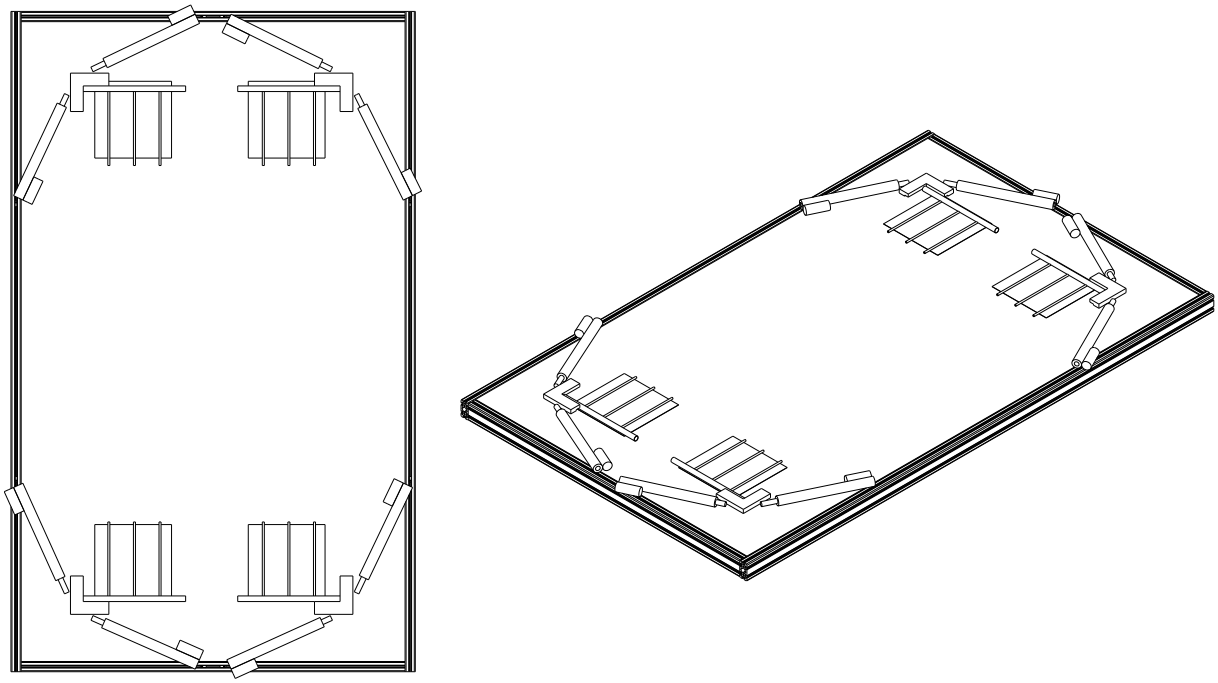
Η συγκράτηση του απορροφητικού φύλλου πραγματοποιείται με αντίστοιχο σύστημα. Σε αυτή την περίπτωση αρκούν μόνο «στοπ» σε δύο, κάθετες μεταξύ τους, πλευρές του φύλλου. Ενδεικτικά έχει τοποθετηθεί ένα σε κάθε πλευρά ενώ μπορούν να τοποθετηθούν περισσότερα αν κριθεί απαραίτητο. Δεν υπάρχει κίνδυνος σύγκρουσης των «στοπ» των δύο πλευρών μεταξύ τους και κινούνται ανεξάρτητα μεταξύ τους. Το μήκος των περιστρεφόμενων προφίλ χρειάζεται παρόλα αυτά να προσδιοριστεί σε συνδυασμό με τα μήκη των υπολοίπων «στοπ» του πλέγματος καθώς μοιράζονται τον ίδιο χώρο. Οι απολήξεις τους πρέπει να έχουν χαμηλό ύψος ώστε να μπορούν να εφάπτονται στο φύλλο χωρίς να εφάπτονται στις πλευρές των header. Τόσο εύρος διαστάσεων που μπορούν να φορτωθούν όσο και η ροπή συγκράτησης αυξάνονται γραμμικά με το μήκος των περιστρεφόμενων προφίλ (μοχλοβραχίονες). Συνεπώς οι απαιτήσεις για μεγαλύτερο δυνατό εύρος διαστάσεων και χαμηλότερο δυνατό ύψος στοιχείων είναι αντικρουόμενες, λόγω αύξησης του μεγέθους των κινητήρων για μεγαλύτερη ισχύ.



Σχήμα 13: Σύστημα τοποθέτησης του απορροφητικού φύλλου της πρώτης σχεδιαστικής λύσης.

Η δεύτερη λύση που εξετάστηκε απεικονίζεται στο σχήμα 14. Σε αυτή τη διάταξη η μετα-

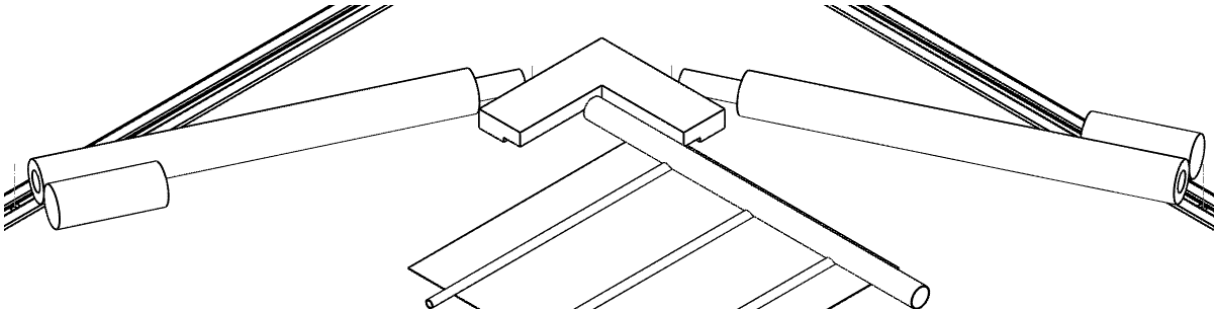
κίνηση κάθε γωνία-«στοπ» γίνεται με χρήση ζεύγους γραμμικών επενεργητών. Για επαρκές μήκος stroke (λ.χ. 400 mm) μπορεί να καλυφθεί αρκετά μεγάλο εύρος διαστάσεων, εφόσον εξασφαλιστεί ότι δεν εμποδίζεται η κίνηση μόλις οι επενεργητές διέλθουν από το ιδιόμορφο σημείο στο οποίο βρίσκονται ομοαξονικά τοποθετημένοι. Πάνω στις γωνίες μπορούν να βρίσκονται συγκρατητές ελάσματα ή quick clamp αλλά δεν έχουν απεικονισθεί καθώς αυτή η διάταξη δεν παρέχει επαρκή αντίσταση σε κατακόρυφη κίνηση, όπως θα εξηγηθεί και στη συνέχεια. Επιπλέον δεν υπάρχει επαρκής χώρος για να τοποθετηθούν τα «στοπ» των δύο πλευρών του φύλλου τα οποία υποχρεωτικά χρειάζονται ξεχωριστό μηχανισμό.



Σχήμα 14: Δεύτερη σχεδιαστική λύση διάταξης ιδιοσυσκευής συγκράτησης. Απλοποιημένα απεικονίζονται με μπλε χρώμα οι γραμμικοί βηματικοί κινητήρες, με πράσινο οι γωνίες-«στοπ» του πλέγματος και του φύλλου.

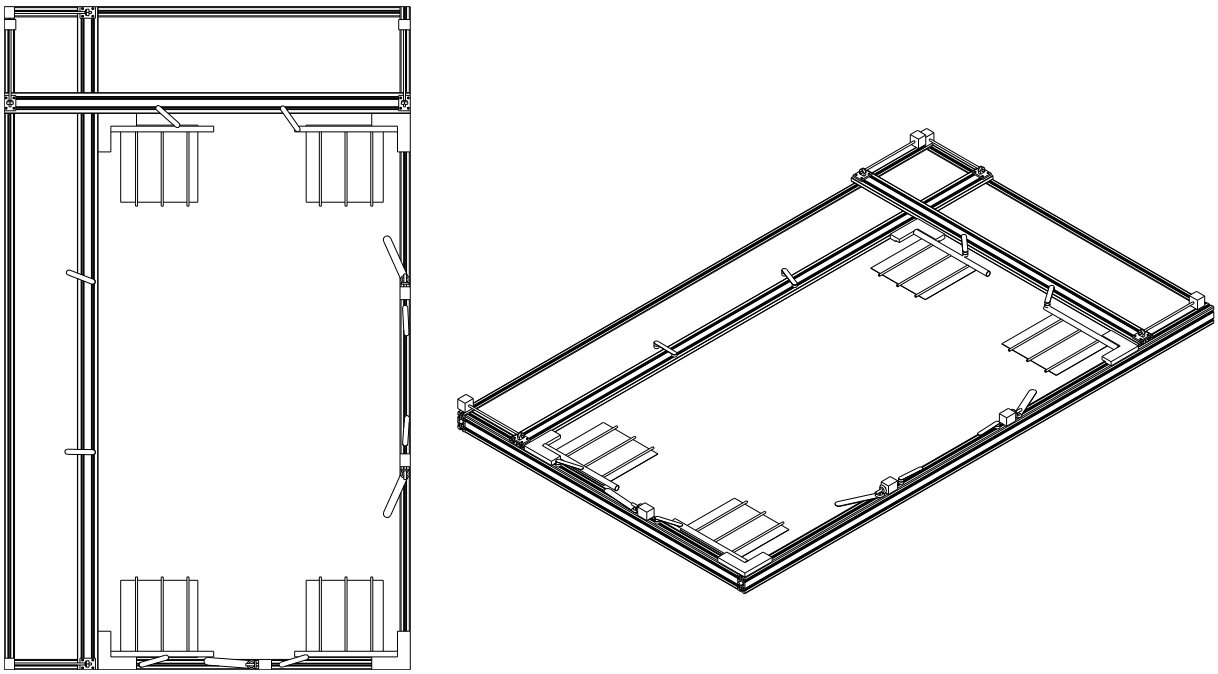
Σε αυτή την περίπτωση σε κάθε άκρη header αντί για δύο ανεξάρτητα κινούμενες επιφάνειες «στοπ» υπάρχει μία ενιαία γωνία. Η δυνατότητα κίνησής της στις δύο διευθύνσεις του επιπέδου επιτυγχάνεται με χρήση ζεύγους γραμμικών ηλεκτρικών επενεργητών. Το ζεύγος είναι απαραίτητο καθώς υπάρχουν δύο βαθμοί ελευθερίας κίνησης. Οι επενεργητές περιστρέφονται ελεύθερα πάνω στα προφίλ της τράπεζας ώστε κατά την έκταση ή συστολή τους να λαμβάνεται το κατάλληλο (μοναδικό εκατέρωθεν της διαγωνίου μεταξύ τους) ζεύγος γωνιών για κάθε θέση. Παρόλο που ο υπολογισμός των κατάλληλων κινήσεων για τοποθέτηση είναι σχετικά απλός, δύο ζητήματα που χρήζουν ανάλυσης είναι η διατήρηση του προσανατολισμού κάθε γωνίας κατά τη μετακίνηση και η αποφυγή του ιδιόμορφου σημείου στο οποίο οι επενεργητές είναι ευθυγραμμισμένοι μεταξύ τους και είναι αδύνατη η περιστροφική κίνηση μέσω έκτασης ή συστολής. Επίσης, ανάλογα με την τελική επιλογή του μήκους των επενεργητών θα εξαρτηθεί ο διαθέσιμος χώρος για τοποθέτηση περιστροφικού «στοπ» φύλλων παρόμοια με την προηγούμενη λύση.

Η τρίτη λύση που εξετάστηκε φαίνεται στο σχήμα 16. Σε αυτή τη διάταξη η μία γωνία-«στοπ» είναι σταθερά τοποθετημένη στην μία γωνία της τράπεζας ενώ οι υπόλοιπες βρίσκονται στερεωμένες πάνω σε γραμμικά κινούμενα προφίλ αλουμινίου. Το κάθε προφίλ είναι παράλληλο στη μία πλευρά της τράπεζας και κινείται γραμμικά με δύο βηματικούς κινητήρες με ατέρμονες



Σχήμα 15: Σύστημα γωνίας κινούμενης από δύο γραμμικούς επενεργητές της δεύτερης σχεδιαστικής λύσης

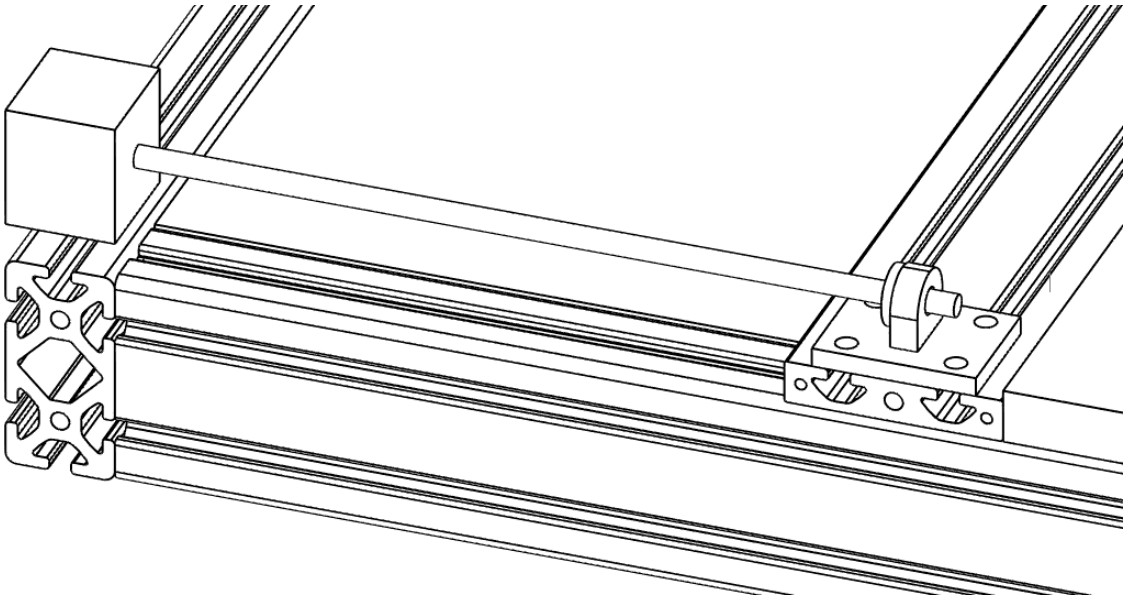
κοχλίες (lead screw) στα άκρα τους. Εδώ χρησιμοποιήθηκαν περιστρεφόμενοι συγκρατητές ελάσματα για την κατακόρυφη συγκράτηση των header καθώς εξαιτίας των προφίλ δεν υπάρχει μεγάλο περιθώριο χώρου καθ΄ ύψος. Τα «στοπ» των πλευρών του φύλλου κινούνται περιστροφικά με χρήση βηματικών κινητήρων και ζεύγους bevel gears με λειτουργία ανάλογη με αυτήν της πρώτης λύσης. Η τοποθέτησή τους καθώς και των συγκρατητών πάνω στις μπάρες είναι ενδεικτικές καθώς εξαρτώνται από το εύρος πιθανών διαστάσεων συλλέκτη. Οι βηματικοί κινητήρες με lead screw που απεικονίζονται διαθέτουν μήκος διαδρομής 300μμ. Για να γίνει εμφανές ο τρόπος που θα ελεγχθεί η σύγκρουση με τις κεφαλές laser απεικονίζονται κινούμενες πάνω στους δύο ακραίους riser, οι γεωμετρίες με τις μέγιστες διαστάσεις των κεφαλών laser.



Σχήμα 16: Τρίτη σχεδιαστική λύση διάταξης ιδιοσυσκευής συγκράτησης. Απλοποιημένα απεικονίζονται με μπλε χρώμα οι βηματικοί κινητήρες με lead screw, με ροζ οι συγκρατητές ελάσματα για τους header, με κίτρινο τα «στοπ» των πλευρών του φύλλου, και με γκρι τα προφίλ αλουμινίου. Με σκούρο γκρι απεικονίζονται οι κεφαλές του laser όταν κινούνται πάνω στους δύο ακραίους riser.

Η μετακίνηση κάθε μπάρας από προφίλ αλουμινίου πραγματοποιείται με χρήση ατέρμονα κο-

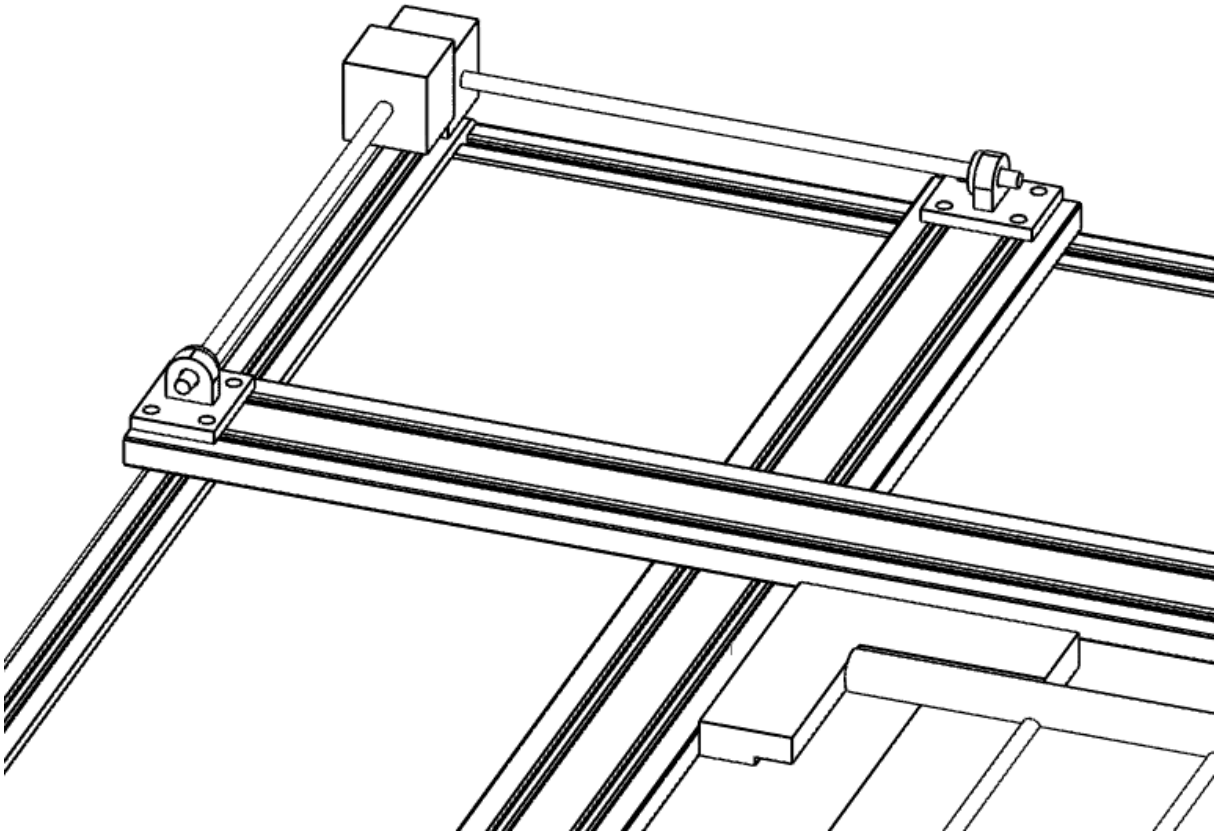
χλία σε κάθε άκρο της. Οι κινητήρες και οι κοχλίες πρέπει να βρίσκονται κατά μήκος των προφίλ της τράπεζας ώστε να δεσμεύεται κατά το δυνατόν λιγότερος διαθέσιμος χώρος για κίνηση των υπόλοιπων στοιχείων. Η ευθυγράμμιση των μπαρών εξασφαλίζεται με την συγχρονισμένη κίνηση του ζεύγους κινητήρων της και συνεπώς δεν είναι απαραίτητη η ύπαρξη οδηγών. Επειδή οι μπάρες ολισθαίνουν πάνω στην τράπεζα, ακόμα και αν επαρκεί η ισχύς των κινητήρων, θα πρέπει να ελαχιστοποιηθούν οι τριβές μεταξύ μπαρών και τράπεζας ώστε να μην σημειώνεται φθορά και ασυνέχειες στην κίνηση εξαιτίας αυτής. Επιπλέον είναι απαραίτητη η συγκράτηση των κοχλιών στα άκρα τους με ρουλεμάν.



Σχήμα 17: Σύστημα κίνησης της μπάρας με ατέρμονα κοχλία της τρίτης σχεδιαστικής λύσης.

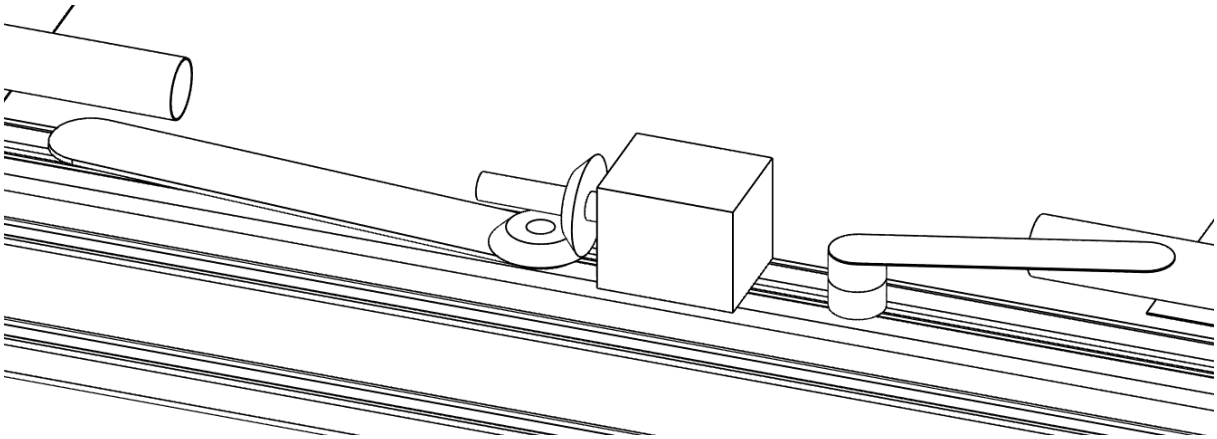
Σε αυτόν τον σχεδιασμό η μία γωνία είναι πλήρως στερεωμένη στην τράπεζα, οι δύο κινούνται σε μία διεύθυνση πάνω σε μία μπάρα και η τέταρτη κινείται πάνω στο σημείο σύνδεσης των δύο μπαρών. Για να μην εμφανίζονται μεγάλα βέλη κάμψης επιλέχθηκε η μακρύτερη μπάρα να ολισθαίνει πάνω στην τράπεζα και η κάθετη στηρίζεται πάνω της. Στο σημείο σύνδεσής τους καθώς και στα σημεία πρόσδεσης των κινητήρων και των υπολοίπων στοιχείων στα προφίλ της τράπεζας θα χρειαστεί να τοποθετηθούν σχεδιασμένα τεμάχια και μετατροπείς. Ενδεικτικά έχουν σχεδιαστεί τα στοιχεία σύνδεσης των μπαρών με τους ατέρμονες. Στις σχεδιασμένες γωνίες χρειάζεται να υπάρχει περιθώριο στο οποίο χωράει το τμήμα του φύλλου που προεξέχει πέρα από την πλευρά του header.

Η κατακόρυφη συγκράτηση των header είναι εφικτή με χρήση περιστρεφόμενων ελασμάτων πάνω σε αποστάτες. Αυτά βρίσκονται βιδωμένα σε σταθερές θέσεις πάνω στα προφίλ της τράπεζας ή στις μπάρες ώστε να είναι διαθέσιμα για κάθε περίπτωση προϊόντος. Το μήκος και η διατομή (πλάτος και πάχος) των ελασμάτων χρειάζεται να προσδιορισθούν σε συνδυασμό με την τελική τοποθέτηση των υπολοίπων στοιχείων καθώς παρόλο που η δύναμη συγκράτησης χρειάζεται να είναι επαρκώς μεγάλη δεν θα πρέπει να υπερβαίνει ένα άνω όριο ώστε να μην φθείρει τον σωλήνα κατά την τοποθέτηση. Τα περιστρεφόμενα «στοπ» φύλλων λειτουργούν κατά τρόπο παρόμοιο με αυτά της πρώτης λύσης. Εδώ η διαφορά είναι ότι η στήριξη πραγματοποιείται με τα άκρα των στοπ τα οποία είναι κυκλικά. Η κίνηση δίνεται από κινητήρα πάνω στο προφίλ της τράπεζας και μέσω μειωτήρα κάθετης γωνίας. Ο μειωτήρας είναι αποτελεσματική λύση αφενός επειδή εξοικονομείται κατακόρυφος χώρος τοποθετώντας τους



Σχήμα 18: Σύστημα γωνίας κινούμενης από τις δύο μπάρες της τρίτης σχεδιαστικής λύσης

κινητήρες οριζόντια και αφετέρου διότι αυξάνουν τη ροπή συγκράτησης χωρίς να χρειάζεται να τοποθετηθούν επιπλέον φρένα.



Σχήμα 19: Σύστημα τοποθέτησης του απορροφητικού φύλλου και αριστερά του συγκρατητής πλέγματος σωλήνων της τρίτης σχεδιαστικής λύσης.

Συγκρίνοντας τις τρεις λύσεις προκύπτουν τα προτερήματα της κάθε μίας έναντι των υπολοίπων. Και οι τρεις λύσεις ικανοποιούν τον πρώτο περιορισμό χώρου και τον τρίτο περιορισμό αφού στερεώνονται πάνω σε τράπεζα. Με κατάλληλη τοποθέτηση κάποιων στοιχείων τους και επιλογή διαστάσεων ικανοποιούν και τον δεύτερο περιορισμό σύγκρουσης με το laser, σε βάρος ταυτόχρονα της τέταρτης απαίτησης για το μέγιστο εύρος διαστάσεων. Επιπλέον και

οι τρεις λύσεις ικανοποιούν την δεύτερη απαίτηση για την επίπεδη συγκράτηση του πλέγματος. Η πρώτη και η τρίτη λύση ικανοποιούν και την πρώτη απαίτηση για την συγκράτηση του φύλλου ενώ η δεύτερη θα χρειαζόταν ριζική τροποποίηση για να την εξυπηρετήσει. Αξίζει να σημειωθεί ότι η δεύτερη λύση είναι σημαντικά απλούστερη έναντι των υπολοίπων (και θα μπορούσε να ήταν ακόμα περισσότερο αν η μία γωνία ήταν στερεωμένη) ενώ η πρώτη είναι η συνθετότερη. Επίσης παρόλο που σε αυτόν τον πρώιμο σχεδιασμό παραλήφθηκαν οι συνδέσεις μεταξύ των προφίλ ή των κινητήρων είναι εμφανές ότι απαιτείται επιπλέον κατακόρυφος χώρος για αυτές της πρώτης λύσης όπου όλες οι κινήσεις είναι περιστροφικές. Σημαντικός παράγοντας είναι η στιβαρότητα και η αντίσταση σε κατακόρυφη κίνηση που προβάλλει η κάθε διάταξη. Η δεύτερη λύση υστερεί σημαντικά σε αυτό το σημείο καθώς θα εμφανίζονταν μεγάλες ροπές στις συνδέσεις των κινητήρων με την τράπεζα αν εφαρμοστεί κατακόρυφη δύναμη στις άκρες τους, εξαιτίας του μεγάλου μήκους από το σημείο εφαρμογής στο σημείο καταπόνησης. Σε μικρότερο βαθμό είναι επίφοβο αυτό στην πρώτη λύση, όπου τα μήκη των προφίλ είναι μικρότερα (στον τωρινό σχεδιασμό) ενώ δεν υπάρχει τέτοιος κίνδυνος στην τρίτη λύση η οποία είναι και η στιβαρότερη από τις τρεις.

Οι παραπάνω παρατηρήσεις συνοψίζονται στον πίνακα 2. Οι τρεις λύσεις βαθμολογήθηκαν συγκριτικά ως προς την απλότητα, και τον βαθμό ικανοποίησης των απαιτήσεων και των περιορισμών. Σύμφωνα με την τελική βαθμολογία επιλέχθηκε η τρίτη λύση προς περαιτέρω ανάπτυξη. Αυτή η λύση είναι σε θέση να ικανοποιήσει όλες τις απαιτήσεις στον καλύτερο δυνατό βαθμό χάρη στην στιβαρότητα της και την απλότητα στο σχεδιασμό.

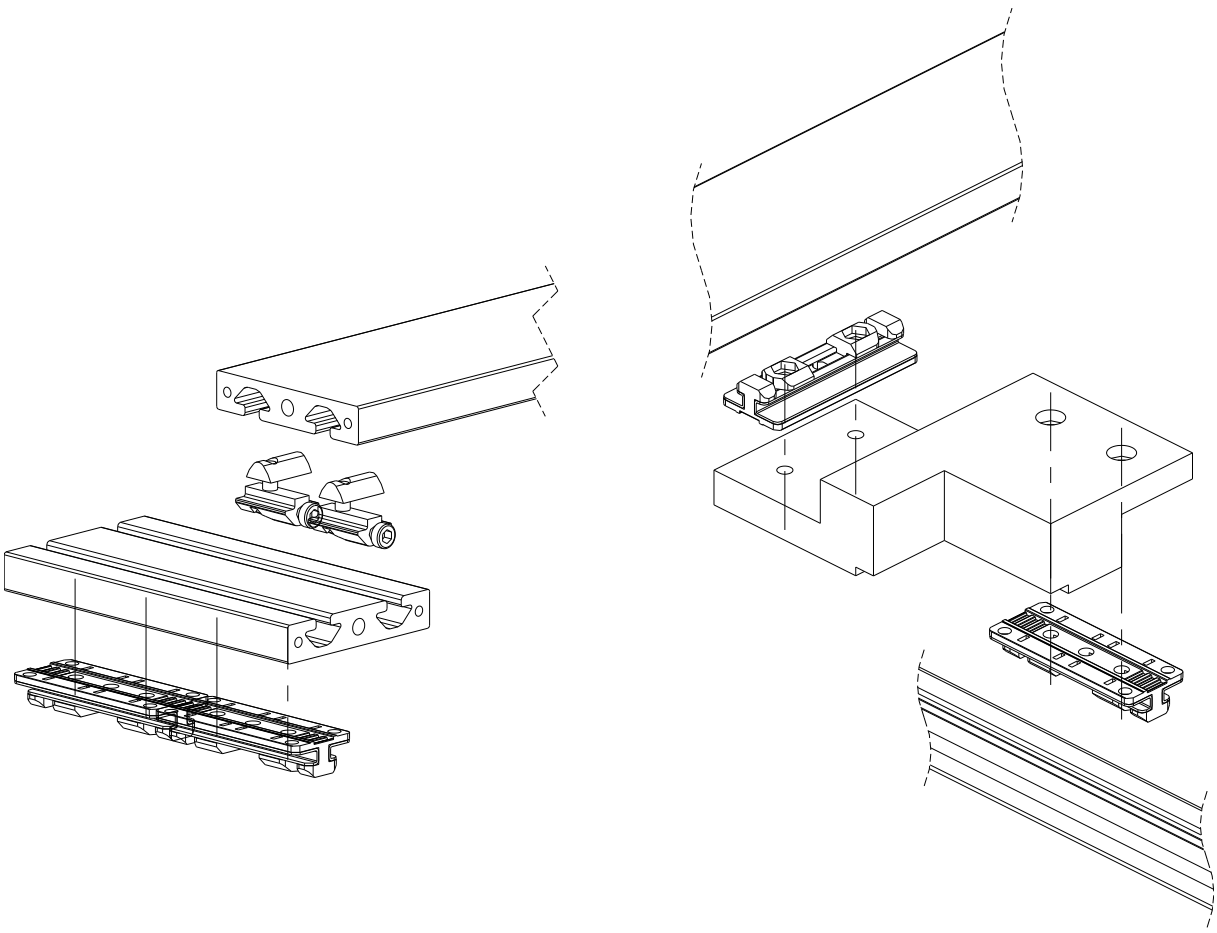
Πίνακας 2: Αξιολόγηση σχεδιαστικών λύσεων.

| Λύση | Ικανοποίηση απαιτήσεων | Ικανοποίηση περιορισμών | απλότητα | Βαθμολογία |
|---|---|---|---|---|
| 1 | •• | ••• | • | 6 |
| 2 | • | •• | ••• | 6 |
| 3 | ••• | ••• | •• | 8 |

## 4.2 Λεπτομερής σχεδιασμός

Στη συνέχεια παρατίθενται οι επιμέρους σχεδιαστικές επιλογές με τις οποίες η επιλεγμένη λύση κατέστη λειτουργική. Χρειάστηκε να τοποθετηθούν ολισθητές σε κάθε σημείο που υπάρχει σχετική κίνηση μεταξύ των προφίλ, δηλαδή στα άκρα κάθε μπάρας και στο σημείο σύνδεσής τους. Έτσι μειώνεται η φθορά των προφίλ και συγχρόνως η τριβή κατά τη γραμμική κίνηση, την οποία καλούνται να αντιμετωπίσουν οι κινητήρες. Οι ολισθητές εισέρχονται στο προφίλ πάνω στο οποίο ολισθαίνουν μέσω γεωμετρίας ανεστραμμένου «T» επενδεδυμένης με πλαστικό. Στα άκρα των κινούμενων μπαρών οι ολισθητές βιδώνονται στις κάτω πλευρές μικρών, κάθετων σε αυτές, τμημάτων προφίλ ενώ στη σύνδεση των δύο μπαρών η οποία γίνεται μέσω του «στοπ-γωνίας» οι ολισθητές βιδώνονται πάνω σε αυτό, όπως φαίνεται στο σχήμα 20.

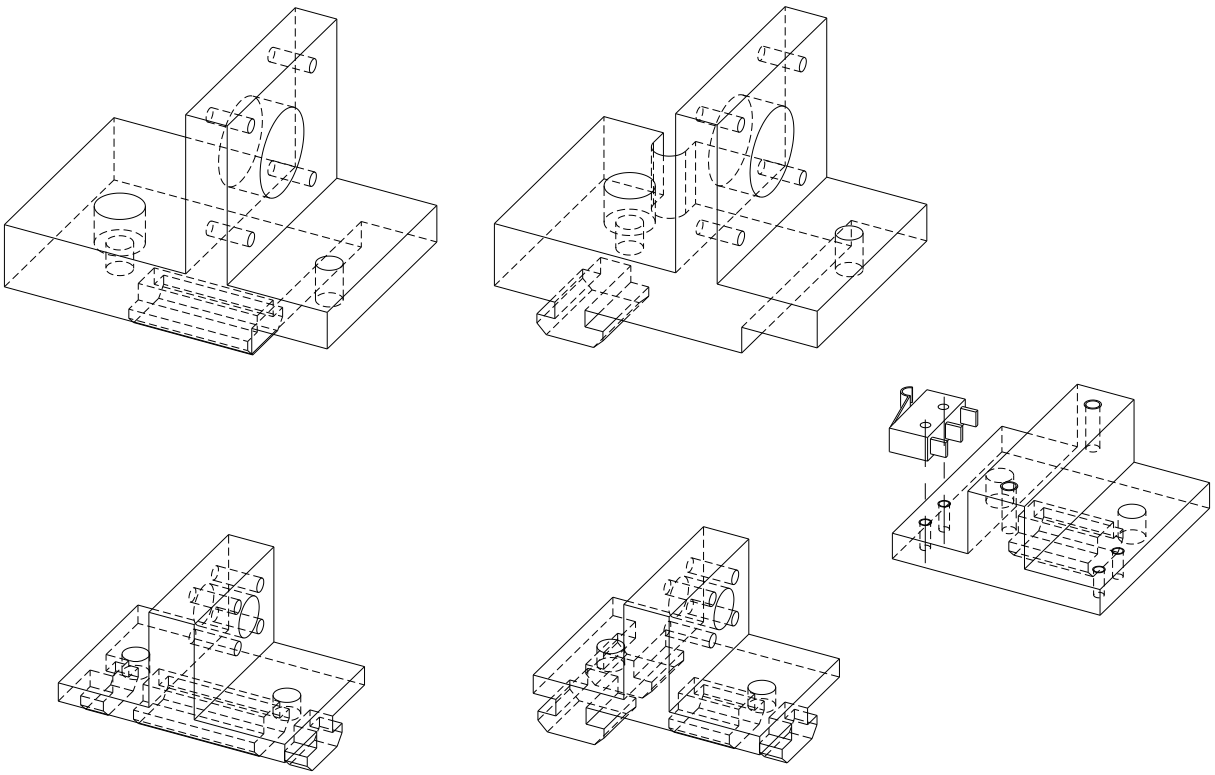
Για να στηριχθούν στις κατάλληλες θέσεις οι κινητήρες σχεδιάστηκαν κατάλληλες βάσεις οι οποίες θα κατασκευαστούν σε κέντρο κατεργασιών. Οι βάσεις των 4 κινητήρων των μπαρών ικανοποιούν δύο βασικές απαιτήσεις: σταθερή σύνδεση με την τράπεζα και κατάλληλη τοποθέτηση (θέση και ευθυγράμμιση) των κινητήρων. Για την πρώτη απαίτηση επιλέχθηκε η χρήση γεωμετρίας ανεστραμμένου T στο κάτω μέρος της βάσης και διάνοιξη οπών ώστε χρησιμοποιηθούν T-slot nuts. Για την δεύτερη απαίτηση επιλέχθηκε η βάση να διαθέτει



Σχήμα 20: Απεικόνιση της σύνδεσης των προφίλ και των ολισθητών στο ένα άκρο της κατακόρυφης μπάρας (αριστερά) και στο σημείο σύνδεσης των δύο μπαρών μέσω της μίας γωνίας-«στοπ» (δεξιά).

κατακόρυφη επιφάνεια με μικρή ανοχή στην καθετότητα και στον προσανατολισμό της, στην οποία θα βιδωθεί ο κινητήρας. Για τη συγκράτηση των τραπεζοειδών κοχλιών στα άκρα τους επιλέχθηκε να χρησιμοποιηθούν κατάλληλα μπλοκ ρουλεμάν τα οποία επίσης τοποθετήθηκαν σε βάσεις. Οι παραπάνω βάσεις απεικονίζονται στο σχήμα 21. Οι ίδιες απαιτήσεις χρησιμοποιήθηκαν και για τις βάσεις των κινητήρων των «στοπ» φύλλων. Για εξοικονόμηση κατακόρυφου χώρου δεν χρησιμοποιήθηκε μειωτήρας ή κιβώτιο right-angle αλλά τοποθετήθηκαν οι κινητήρες με τον άξονα κατακόρυφα. Συνεπώς τέθηκε μικρή ανοχή στην παραλληλία (με την τράπεζα) της οριζόντιας επιφάνειας των βάσεων πάνω στην οποία θα βιδωθεί ο κινητήρας. Στην άκρη του ελάσματος που περιστρέφει ο κινητήρας ακουμπά η μία πλευρά του φύλλου. Η γεωμετρία της άκρης επιλέχθηκε να είναι κυκλική τόσο για να μην καταστρέφεται το φύλλο όσο και για να είναι ακριβής ο τριγωνομετρικός υπολογισμός της γωνίας περιστροφής, όπως θα αναλυθεί στη συνέχεια. Η διάταξη απεικονίζονται στο σχήμα 22.

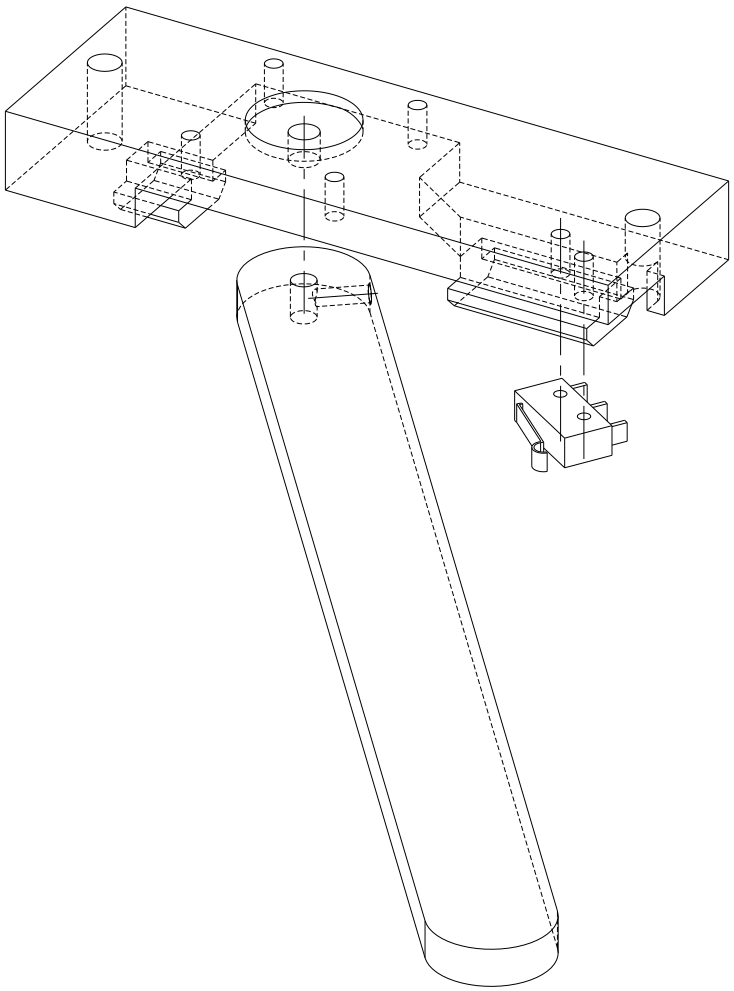
Για να είναι δυνατό να στερεωθούν συλλέκτες με απόσταση κέντρο-κέντρο τουλάχιστον 1200 mm και με μήκος header τουλάχιστον 900 mm έγινε κατάλληλη επιλογή του μήκους των κοχλιών των κινητήρων κάθε μπάρας. Συγκεκριμένα, με τη χρήση ψηφιακού μοντέλου προσδιορίστηκε ότι απαιτούνται κοχλίες μήκους 500 mm για την κατακόρυφη μπάρα και κοχλίες μήκους 1200 mm για την οριζόντια. Η επιλογή αυτή εξαρτάται άμεσα από την τοποθέτηση των υπολοίπων στοιχείων της ιδιοσυσκευής πάνω στα προφίλ της τράπεζας. Πέρα από τον μέγιστο περιορισμό ύψους για την ασφαλή εναλλαγή των τραπεζών, το ύψος των στοιχείων



Σχήμα 21: Από πάνω-αριστερά με ωρολογιακή φορά, η βάση ενός κινητήρα της οριζόντιας μπάρας, η αντίστοιχη βάση της κατακόρυφης μπάρας (με την εγκοπή στην άκρη για προστασία των καλωδίων), η βάση του μπλοκ ρουλεμάν μαζί με τον οριακό διακόπτη, η βάση σύνδεσης με το περικόχλιο του τραπεζοειδή κοχλία και σύνδεσης των καθέτων τμημάτων προφίλ με την κατακόρυφη μπάρα (γεωμετρίες «T» σε δύο διευθύνσεις), και η αντίστοιχη βάση της οριζόντιας μπάρας.

περιορίζει και τον χώρο κίνησης του laser. Επομένως, αφού πρώτα τοποθετήθηκαν όλοι οι κινητήρες με τις βάσεις τους, προσδιορίστηκαν οι ακραίες θέσεις στις οποίες μπορεί να κινείται και να κολλά το laser (και στις δύο διευθύνσεις), και με βάση αυτές προσδιορίστηκε το εύρος κίνησης κάθε μπάρας, δηλαδή τα μήκη των κοχλιών. Ένα επιπλέον πλεονέκτημα της χρήσης κοχλιών (lead screws) εξαιτίας της μετατροπής περιστροφικής σε ευθύγραμμη κίνηση, πέρα από την αύξηση της ακρίβειας θέσης, είναι και η αύξηση της δύναμης συγκράτησης. Καθώς δεν είναι δυνατό να περιστραφούν οι κοχλίες ασκώντας δύναμη στις μπάρες, αυτές διατηρούν τη θέση τους ακόμα και όταν οι κινητήρες δεν ρευματοδοτούνται, όπως κατά τη συγκόλληση.

Πρέπει να σημειωθεί ότι η χρήση κοχλιών συνεπάγεται και αύξηση του απαιτούμενου χρόνου μεταφοράς των μπαρών και συνεπώς του συνολικού χρόνου προετοιμασίας της μηχανής ο οποίος όμως είναι αποδεκτός καθώς σε κάθε περίπτωση δεν υπερβαίνει τα 2 με 3 λεπτά. Λ.χ. για βηματικό κινητήρα μεγέθους NEMA 17, με μικρή ταχύτητα περιστροφής 50 rpm, για ευθύγραμμη μετατόπιση 500 mm χρειάζονται $\frac{500\,\mathrm{mm}}{8\frac{\mathrm{mm}}{\text{περιστρ.}}50\,\mathrm{rpm}} = 80\,\mathrm{s}$. Παράλληλα, η επιλογή βηματικών κινητήρων παρά το χαμηλό κόστος και την επαρκή ακρίβεια θέσης στερείται την δυνατότητα απόλυτης τοποθέτησης. Για αυτό το λόγο στην αρχή κάθε ρύθμισης οι μπάρες και τα «στοπ» χρειάζεται να επιστρέφουν σε μία αρχική ακραία θέση (homing) η οποία αποτελεί αναφορά για τις υπολογισμένες μετατοπίσεις. Επομένως στις βάσεις των μοτέρ



Σχήμα 22: Απεικόνιση της σύνδεσης της βάσης του κινητήρα με το περιστρεφόμενο «στοπ» φύλλου και του οριακού διακόπτη.

τοποθετήθηκαν οριακοί διακόπτες (limit switches) το κλείσιμο των οποίων σηματοδοτεί στο πρόγραμμα του ελεγκτή τη διακοπή της επαναφοράς και την εκκίνηση της απαιτούμενης μετατόπισης.

Πάνω στην τράπεζα χρειάζεται να βρίσκονται συνολικά 8 βηματικοί κινητήρες (4 για τις δύο μπάρες και 4 για δύο ζεύγη «στοπ» φύλλων) και συνολικά 6 οριακοί διακόπτες (ένας για κάθε μπάρα και ένας για κάθε «στοπ» φύλλου). Κάθε κινητήρας συνδέεται μέσω 4 καλωδίων με τον driver (εκτός τράπεζας), και κάθε οριακός διακόπτης χρειάζεται 2 καλώδια. Αυτά τα καλώδια θα βρίσκονται στις εσωτερικές κοιλότητες των προφίλ της τράπεζας για επιπλέον προστασία τους. Για αυτό το λόγο έχουν διαμορφωθεί εγκοπές στις πλευρές μερικών βάσεων των κινητήρων των μπαρών ενώ τα καλώδια των κινητήρων των «στοπ» μπορούν να εισέρχονται στις κοιλότητες απευθείας διαμέσου οπών στα προφίλ. Συνολικά πάνω στην τράπεζα βρίσκονται $8 \cdot 4 + 8 \cdot 2 = 48$ καλώδια διαμέτρου 0.5 mm (κατηγορίας μεγέθους 24AWG) τα οποία χρειάζεται να συνδέονται αυτόματα με τον ελεγκτή και τους drivers κάθε φορά που η τράπεζα φτάνει στην θέση φόρτωσης-εκφόρτωσης. Αυτό μπορεί να επιτευχθεί με χρήση επαφών και συνδέσμων με ελατήρια (spring-loaded contacts and connectors ή «pogo pins», όπως αυτά των σχημάτων 23 και 24. Καθώς επαρκεί ένας σύνδεσμος για κάθε καλώδιο (δυνατότητα διέλευσης έως και 9 A) μπορούν να τοποθετηθούν σε συστοιχία (των $12 \times 4$ με διαστάσεις περίπου $48\,\text{mm} \times 16\,\text{mm}$) στην αριστερή πλευρά της τράπεζας σε κατακόρυφη

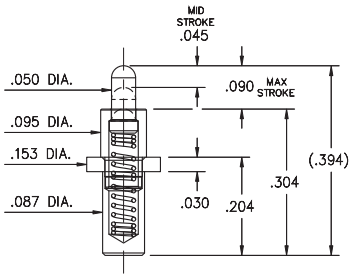
διεύθυνση ώστε να έρχονται ομαλά σε επαφή με τις σταθερές επαφές κατά την ανοδική κίνηση της τράπεζας. Αφού ολοκληρωθεί η ρύθμιση και το φόρτωμα η τράπεζα κινείται οριζόντια και διακόπτεται η επαφή των συνδέσμων.

## PRODUCT NUMBER: 0850-0-15-20-83-14-11-0

**0850-0-15-20-83-14-11-0**

Power spring pin, Solder mount in .090 min. mounting hole. Also available on 32mm wide X 12mm pitch carrier tape: 500 parts per 13" reel

Order as: 0850-0-57-20-83-14-11-0




### DESCRIPTION

Spring-Loaded Power Pin (9A current rating)

**Durability:**

100,000 to 1,000,000 Cycles

**Current Rating:**

9A @ 10°C Temperature Rise

**Contact Resistance:**

20 mΩ Max

**Operating Temperature Range:**

-55/+125°C (discontinuous)

**Vibration:**

No Elect. Discontinuity > 1µs @ 0-200 HZ, 10G

**Shock:**

No Elect. Discontinuity > 1µs @ 50 G

**Mounting Feature:**

Through-Hole Solder Mount

**Mounting Hole:** .090" (2,286mm)

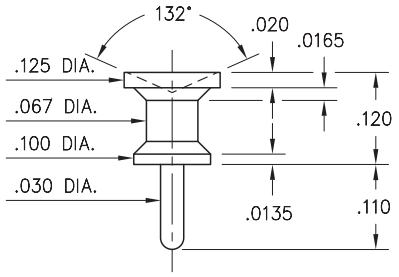
**Packaging:** 15 - Packaged in Bulk

| Shell Plating | Contact Plating | ROHS |
|---|---|---|
| 20 µ" Gold over Nickel | 10 µ" Gold over Nickel | RoHS-2 2011/65/EU |

Σχήμα 23: Τεχνικά χαρακτηριστικά του συνδέσμου με ελατήριο, από τον κατάλογο της Mill-Max.

## PRODUCT NUMBER: 1945-0-00-15-00-00-03-0

**1945-0-00-15-00-00-03-0**
Concave face Target contact
Solder mount in .034 min. mounting hole

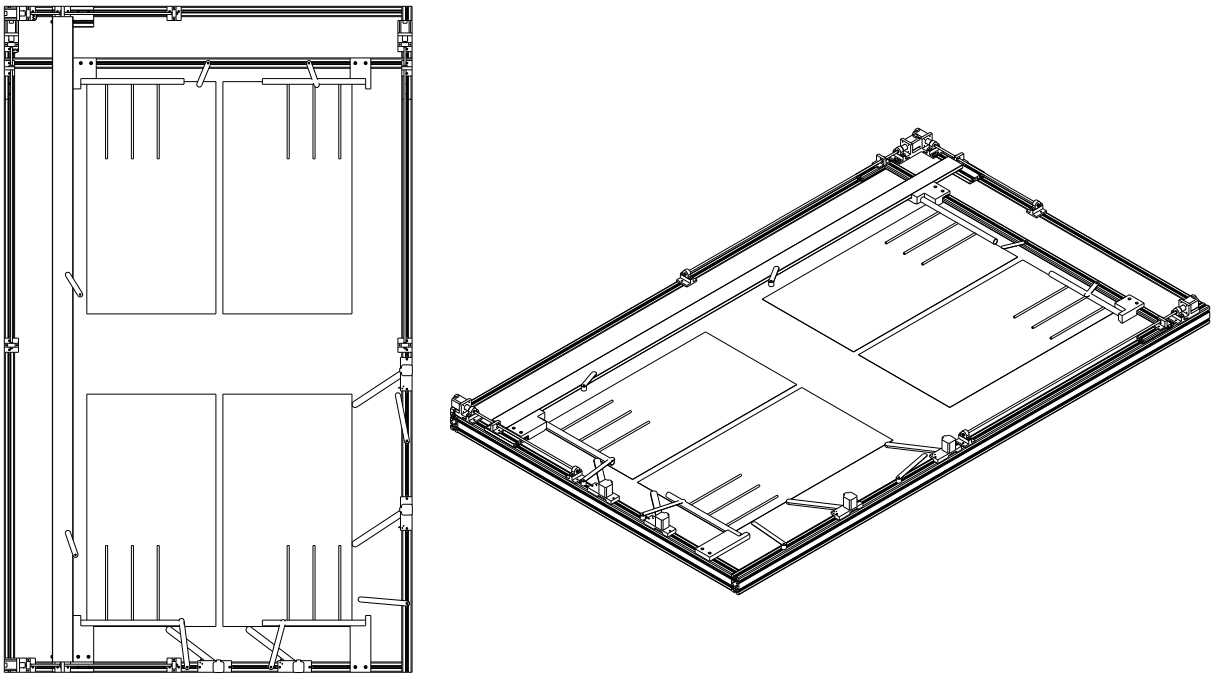



### DESCRIPTION

Target Contact for Spring-Loaded Pin
**Mounting Feature:**
Through-Hole Solder Mount
**Tail Type:** Soldertail
**Tail Diameter:** .030" (0,762mm)
**Packaging:** 00 - Packaged in Bulk

| Shell Plating | Contact Plating | ROHS |
|---|---|---|
| 10 µ" Gold over Nickel | | RoHS-2 2011/65/EU |

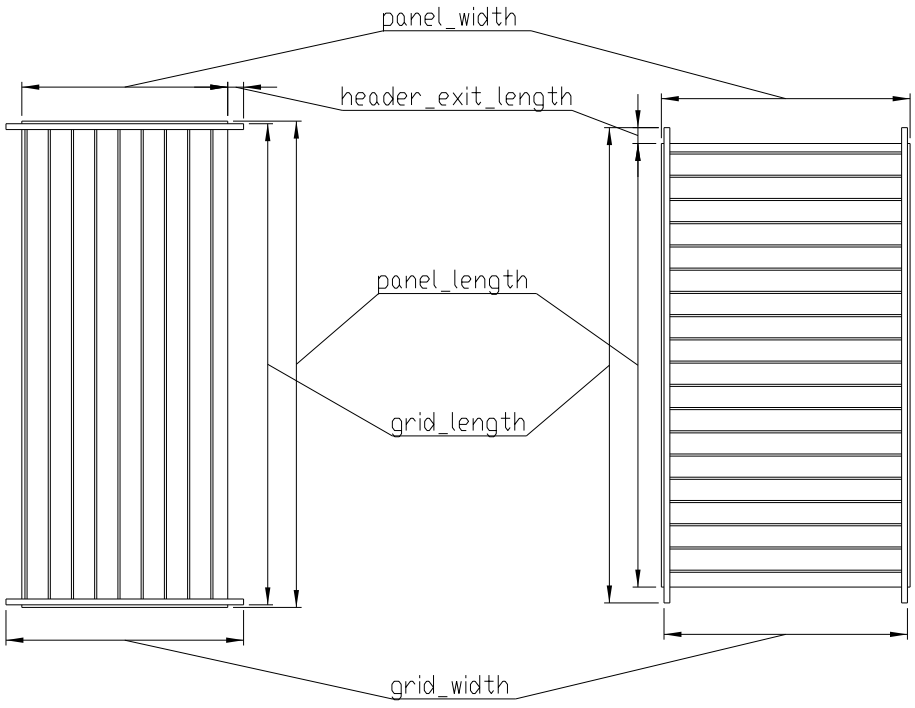
Σχήμα 24: Τεχνικά χαρακτηριστικά της επαφής για τον σύνδεσμο με ελατήριο, από τον κατάλογο της Mill-Max.



Σχήμα 25: Τελικός σχεδιασμός της ιδιοσυσκευής συγκράτησης.

## 4.3 Περιγραφή χρήσης

Η ρύθμιση της ιδιοσυσκευής από τους χειριστές απαιτεί μόνο την εισαγωγή ορισμένων γεωμετρικών χαρακτηριστικών ενός ηλιακού συλλέκτη σε κατάλληλο πρόγραμμα σε Η/Υ ο οποίος επικοινωνεί με τους ελεγκτές των κινητήρων της ιδιοσυσκευής. Ένα τέτοιο πρόγραμμα γράφτηκε σε γλώσσα Python 3.7, με χρήση του πακέτου PySimpleGUI για την δημιουργία του interface, και μεταγλωττίστηκε σε εκτελέσιμο με τον PyInstaller. Οι απαραίτητες κινήσεις που πρέπει να γίνουν για να ρυθμιστεί η διάταξη είναι αυτή της κατακόρυφης μπάρας, αυτή της οριζόντιας, αυτή των δεξιών «στοπ» φύλλου και αυτή των κάτω «στοπ» φύλλου. Για να υπολογιστούν αυτές, ανεξαρτήτως του τύπου συλλέκτη (κατακόρυφος ή οριζόντιος), θα πρέπει να είναι γνωστά η διάμετρος των header, η απόσταση κέντρο-κέντρο των header, το μήκος φύλλου, και το πλάτος φύλλου μαζί με το μήκος ελεύθερης εξόδου header (ή εναλλακτικά το μήκος ενός header μόνο στην περίπτωση που και οι δύο έξοδοι του είναι ανοιχτές). Στο σχήμα 26 απεικονίζονται αυτές οι διαστάσεις για κάθε τύπο συλλέκτη. Οι απαραίτητες μετατοπίσεις υπολογίζονται με αναφορά κάθε οριακό διακόπτη. Για να επιτευχθεί αυτό χρησιμοποιήθηκε ένα σύνολο παραμέτρων οι οποίες προέρχονται από τις διαστάσεις της σχεδιασμένης διάταξης, και παρουσιάζονται στο σχήμα 27. Επιπλέον δίνεται η δυνατότητα εισαγωγής offset για κάθε κινητήρα το οποίο προστίθεται σε κάθε κίνηση με σκοπό την αντιστάθμιση μικρών σφαλμάτων που ενδέχεται να εμφανιστούν στο μέλλον.




Σχήμα 26: Βασικά γεωμετρικά χαρακτηριστικά κατακόρυφου και οριζόντιου συλλέκτη που χρησιμοποιούνται στη ρύθμιση της διάταξης

Οι κινήσεις σε βήματα για κάθε κινητήρα υπολογίζονται σύμφωνα με τις παρακάτω σχέσεις.

- Για κάθε κινητήρα της κατακόρυφης μπάρας:

$$(\text{περιστροφές}) = \frac{\texttt{MAX\_WIDTH} + \texttt{offset} - \texttt{grid\_width}}{\texttt{SCREW\_LEAD}}$$

$$(\text{βήματα}) = \text{floor}\left(\frac{(\text{περιστροφές}) \cdot 360}{\texttt{STEP\_ANGLE}}\right)$$

- Για κάθε κινητήρα της οριζόντιας μπάρας:

$$(\text{περιστροφές}) = \frac{\texttt{MAX\_LENGTH} + \texttt{offset} - \texttt{grid\_length}}{\texttt{SCREW\_LEAD}}$$
$$(\text{βήματα}) = \text{floor}\left(\frac{(\text{περιστροφές}) \cdot 360}{\texttt{STEP\_ANGLE}}\right)$$

- Για κάθε κινητήρα των «στοπ» στις πλευρές του φύλλου:

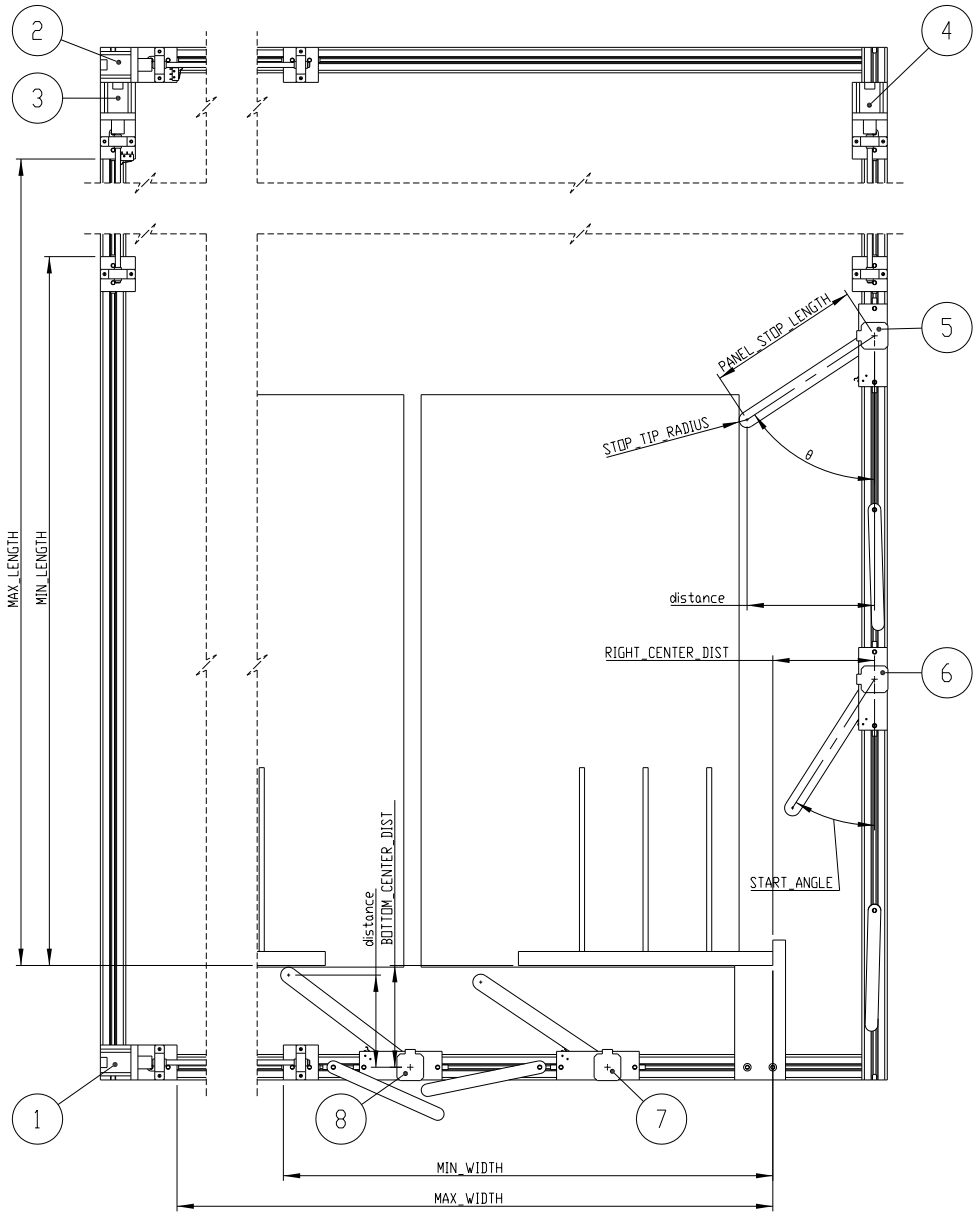
$$(\text{γωνία περιστροφής}) = \arcsin\left(\frac{\texttt{distance}}{\texttt{PANEL\_STOP\_LENGTH}}\right) - \texttt{START\_ANGLE} + \texttt{offsets}$$
$$(\text{βήματα}) = \text{floor}\left(\frac{(\text{γωνία περιστροφής})}{\texttt{STEP\_ANGLE}}\right)$$

Όπου για τα δεξιά «στοπ» η απόσταση `distance` εξαρτάται από την απόσταση `RIGHT_CENTER_DISTANCE` και για τα κάτω από την `BOTTOM_CENTER_DISTANCE`. Τα υπολογισμένα βήματα εξάγονται σε αρχείο κειμένου με την προοπτική να δοθούν ως είσοδος στον ελεγκτή PLC που θα χρησιμοποιηθεί.

Η διεπιφάνεια στην οποία εισάγονται οι πληροφορίες φαίνεται στην εικόνα 11 και ο κώδικας βρίσκεται στο παράρτημα Β. Στο κεντρικό παράθυρο υπάρχουν πεδία εισαγωγής για κάθε γεωμετρικό χαρακτηριστικό και για ένα όνομα ή κωδικό για τον συγκεκριμένο συλλέκτη. Όπως αναλύθηκε ήδη, ζητούνται μόνο οι διαστάσεις με βάσει τις οποίες θα προσδιοριστεί ο χώρος που καταλαμβάνουν το πλέγμα των σωλήνων και το απορροφητικό φύλλο, στον οποίο προσαρμόζεται τελικά η διάταξη. Συγχρόνως, ζητείται και ο τύπος του συλλέκτη (οριζόντιος ή όχι) καθώς οι ίδιες διαστάσεις σε κάθε περίπτωση χρησιμοποιούνται με διαφορετικό τρόπο κατά τον υπολογισμό.

Κάτω από τα πεδία υπάρχει κενή περιοχή στην οποία εμφανίζονται προειδοποιητικά μηνύματα κάθε φορά που εκτελείται μία ενέργεια (λ.χ. εισαγωγή τιμής, αποθήκευση) σε περίπτωση μη έγκυρου συνδυασμού τιμών. Σύμφωνα με τις παραμέτρους που ορίστηκαν προηγουμένως μπορεί να εμφανιστεί προειδοποίηση για τις εξής περιπτώσεις:

- *Μη έγκυρη τιμή.*
  Εμφανίζεται στην περίπτωση που κάποιο πεδίο αριθμητικής τιμής δεν περιέχει αριθμό.
- *Επιτρέπονται μόνο θετικοί αριθμοί*
  Εμφανίζεται στην περίπτωση που σε κάποιο πεδίο αριθμητικής τιμής έχει εισαχθεί μη θετικός αριθμός (0 ή αρνητικός).
- *Το μήκος/πλάτος του συλλέκτη είναι μεγαλύτερο/μικρότερο από αυτό που εξυπηρετεί η διάταξη.*
  Εμφανίζεται στην περίπτωση που κάποιο από το μήκος ή το πλάτος του συλλέκτη είναι μεγαλύτερο από το μέγιστο που μπορεί να λάβει η διάταξη. Στην περίπτωση κατακόρυφου συλλέκτη ως μήκος ορίζεται η απόσταση κέντρο-κέντρο + διάμετρος header και ως πλάτος το πλήρες μήκος ενός header με ανοιχτές εξόδους. Στην περίπτωση του οριζόντιου αυτά ορίζονται αντίστροφα.
- *Οι ρυθμίσεις αποθηκεύτηκαν στο αρχείο ...*
  Εμφανίζεται μόλις αποθηκευτούν επιτυχημένα οι υπάρχουσες διαστάσεις συλλέκτη σε εξωτερικό αρχείο ′`.cfg`′.




Σχήμα 27: Γεωμετρικές παράμετροι της διάταξης. Σε κύκλο βρίσκονται οι αριθμοί κάθε κινητήρα.

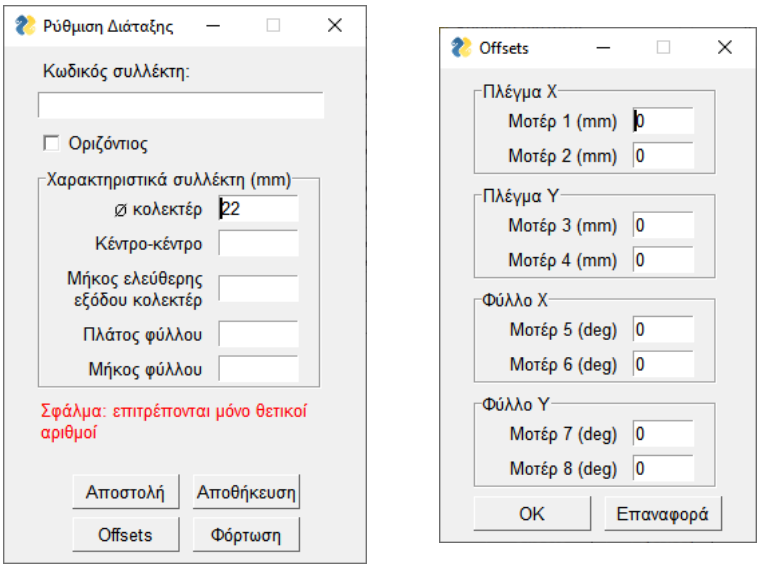
- *Οι ρυθμίσεις φορτώθηκαν από το αρχείο ...*
  Εμφανίζεται μόλις φορτωθούν επιτυχημένα οι αποθηκευμένες διαστάσεις συλλέκτη από εξωτερικό αρχείο ′`.cfg`′.

Παρέχεται η δυνατότητα αποθήκευσης των υπαρχόντων τιμών όλων των πεδίων και επαναφόρτωσής τους μέσω των αντίστοιχων κουμπιών και των εμφανιζόμενων παραθύρων πλοήγησης. Τα δεδομένα αποθηκεύονται σε απλό αρχείο κειμένου, με κατάληξη ′`.cfg`′, και συνεπώς μπορούν να τροποποιηθούν ή να ελεγχθούν εκ των υστέρων από έναν χειριστή. Για να είναι ευανάγνωστα αποθηκεύονται με την εξής μορφή:

```
# περιγραφικό σχόλιο
χαρακτηριστικό = αριθμός
```

Με το κουμπί «Offsets» εμφανίζεται το παράθυρο εισαγωγής των offset στο οποίο υπάρχουν

πεδία εισαγωγής τιμών για κάθε έναν κινητήρα ξεχωριστά. Οι τιμές αποθηκεύονται διαρκώς (σε εξωτερικό αρχείο κατάληξης ΄.dat΄ στην τοποθεσία του εκτελέσιμου) και ανακαλούνται αυτόματα κάθε φορά που εμφανίζεται το παράθυρο. Η μορφή αποθήκευσης είναι η ίδια με την προαναφερθείσα. Αυτές οι τιμές χρησιμοποιούνται στους υπολογισμούς κάθε φορά που πιέζεται το κουμπί «Αποστολή» στο κεντρικό παράθυρο. Η συνεισφορά τους είναι θετική είτε ως γραμμική μετατόπιση είτε ως γωνιακή. Με το κουμπί «Επαναφορά» επαναφέρονται οι αρχικές τιμές από το άνοιγμα του παραθύρου. Με το κουμπί «Αποστολή» στο κεντρικό παράθυρο εξάγεται το αρχείο κειμένου με τον αριθμό βημάτων (και τον προσανατολισμό) για κάθε κινητήρα. Η επικοινωνία αυτού του προγράμματος υπολογισμού και του προγράμματος κίνησης του μικρο-ελεγκτή γίνεται με αυτό το αρχείο.




Εικόνα 11: Κεντρικό παράθυρο και παράθυρο με offsets του προγράμματος ρύθμισης της ιδιοσυσκευής.

# 5 Αξιολόγηση

## 5.1 Μέθοδος αξιολόγησης

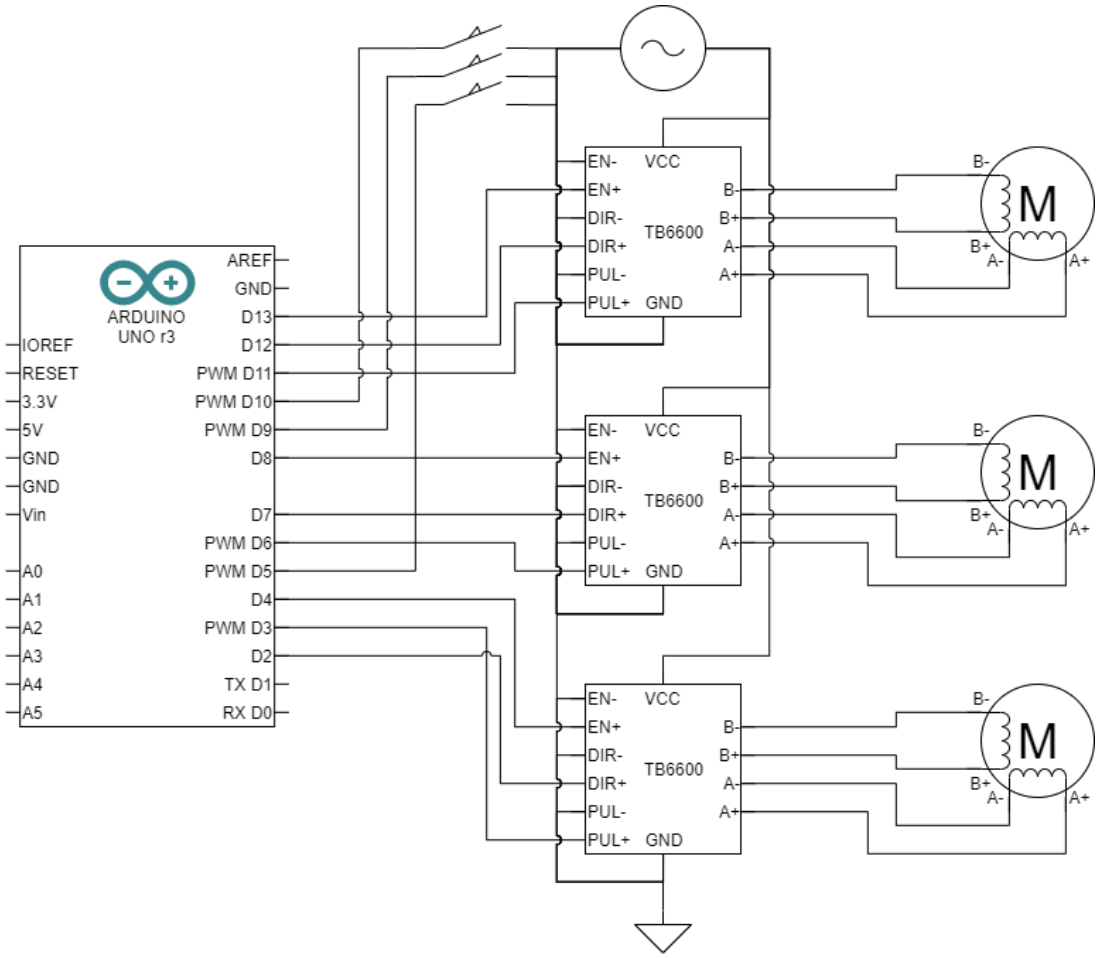
Για να αποδειχθεί ότι ο σχεδιασμός ικανοποιεί τους περιορισμούς και τις απαιτήσεις πραγματοποιήθηκε αξιολόγησή του μέσω δοκιμής. Κατασκευάστηκε η διάταξη της κατακόρυφης μπάρας της οποίας η κίνηση είναι παρόμοια με της οριζόντιας, και ενός «στοπ» φύλλου καθώς και τα 4 κινούνται με τον ίδιο τρόπο. Συνεπώς τα σχεδιασμένα τεμάχια κατασκευάστηκαν στο κέντρο κατεργασιών του εργαστηρίου Τεχνολογίας των Κατεργασιών της σχολής ενώ οι 3 κινητήρες, τα προφίλ αλουμινίου, και οι σύνδεσμοι αγοράστηκαν. Επιπλέον αγοράσθηκαν και οδηγητές (drivers) για τους κινητήρες για να κινηθούν ελεγχόμενοι από μικρο-ελεγκτή.

Για τον έλεγχο των διαστασιολογικών περιορισμών μετρήθηκε το υψηλότερο σημείο και ελέγχθηκε η ύπαρξη προεξοχών πέρα από τις πλευρές της τράπεζας. Για την δοκιμή της κίνησης των μελών επιλέχθηκε να χρησιμοποιηθεί μικρό πρόγραμμα με τον μικρο-ελεγκτή Arduino, το οποίο βρίσκεται στο παράρτημα Γ με την συνδεσμολογία που φαίνεται στο σχήμα 28. Ένα δομικό διάγραμμα του προγράμματος φαίνεται στο σχήμα 29. Στις παραμέτρους του προγράμματος εισάγεται μόνο η συνδεσμολογία των pins των δριερ για τα σετ κινητήρων κάθε κινούμενου μέλους και τους οριακούς διακόπτες τους. Κατά την εκτέλεση, μετά τη μεταφορά στην αρχική θέση (homing) ο χρήστης εισάγει τον επιθυμητό αριθμό βημάτων ο οποίος λαμβάνεται από το πρόγραμμα υπολογισμού. Μέσω των driver ο ελεγκτής πραγματοποιεί τις κινήσεις. Για να λειτουργούν οι κινητήρες κατά το δυνατόν ομαλότερα, καθώς οι μετατοπίσεις είναι αρκετά μικρές και οι συνολικοί χρόνοι μετακίνησης αποδεκτοί, χρησιμοποιήθηκαν μικρές ταχύτητες περιστροφής των κινητήρων, 50rpm για την μπάρα και 5rpm για τα «στοπ». Δεν χρησιμοποιήθηκε micro-stepping.

## 5.2 Δοκιμή και αποτελέσματα

Η συναρμολόγηση έγινε πάνω σε υπάρχον πλαίσιο από προφίλ αλουμινίου ίδιων διαστάσεων με την τράπεζα που χρησιμοποιείται. Η συναρμολογημένη διάταξη φαίνεται στις εικόνες 12, 13, και 14. Το ψηλότερο σημείο της αποτελεί η πίσω πλευρά του κινητήρα του «στοπ» φύλλου, ίσο με 65mm, το οποίο είναι χαμηλότερο από τον περιορισμό των 80mm. Για την στερέωση των στοιχείων πάνω στα προφίλ χρησιμοποιήθηκαν T-slot nuts με κοχλίες M6 διαφόρων μηκών, ενώ για τα μπλοκ ρουλεμάν και τους κινητήρες χρησιμοποιήθηκαν M4 και M3 αντίστοιχα. Χάρη στην πιστότητα των διαστάσεων των βάσεων των κινητήρων της μπάρας εξασφαλίζεται η παραλληλία τους και η ίση απόσταση τους από την άκρη της τράπεζας. Κατά τη συναρμολόγηση δόθηκε προσοχή στην ευθυγράμμιση των ατερμόνων κοχλιών και στην εξάλειψη στρέβλωσης και κάμψης τους μέσω της τοποθέτησης των μπλοκ ρουλεμάν και της βάσης του κινητήρα πριν αυτά βιδωθούν. Επιπλέον επιδιώχθηκε η κατά το δυνατόν ομαλότερη κίνηση της μπάρας φροντίζοντας να είναι ευθυγραμμισμένοι οι ολισθητές κάθε άκρου της και καθαρές οι επιφάνειες των προφίλ στις οποίες ολισθαίνουν.

Κατά τη δοκιμή κίνησης της μπάρας πραγματοποιήθηκε homing και τελική μετακίνηση σε διάφορες θέσεις καθ΄ όλο το μήκος διαδρομής. Η μπάρα κινήθηκε χωρίς να σταματήσει σε κάποιο σημείο ή να «χαθούν» βήματα των κινητήρων και η ακρίβεια κίνησης επαληθεύθηκε με μέτρηση. Η ευθυγράμμιση της μπάρας εξασφαλίζεται μετά από κάθε επιστροφή της στην αρχική θέση χάρη στη χρήση οριακού διακόπτη σε κάθε κινητήρα. Περιοριστικό παράγοντα αποτέλεσε ο μικρο-ελεγκτής εξαιτίας της χαμηλής υπολογιστικής ισχύος του καθώς η ταχύτητα περιστροφής κατά την επιστροφή ήταν σημαντικά χαμηλότερη. Αυτό συνέβαινε διότι έπειτα από κάθε ένα βήμα κατά την επιστροφή εκτελείται έλεγχος της κατάστασης των

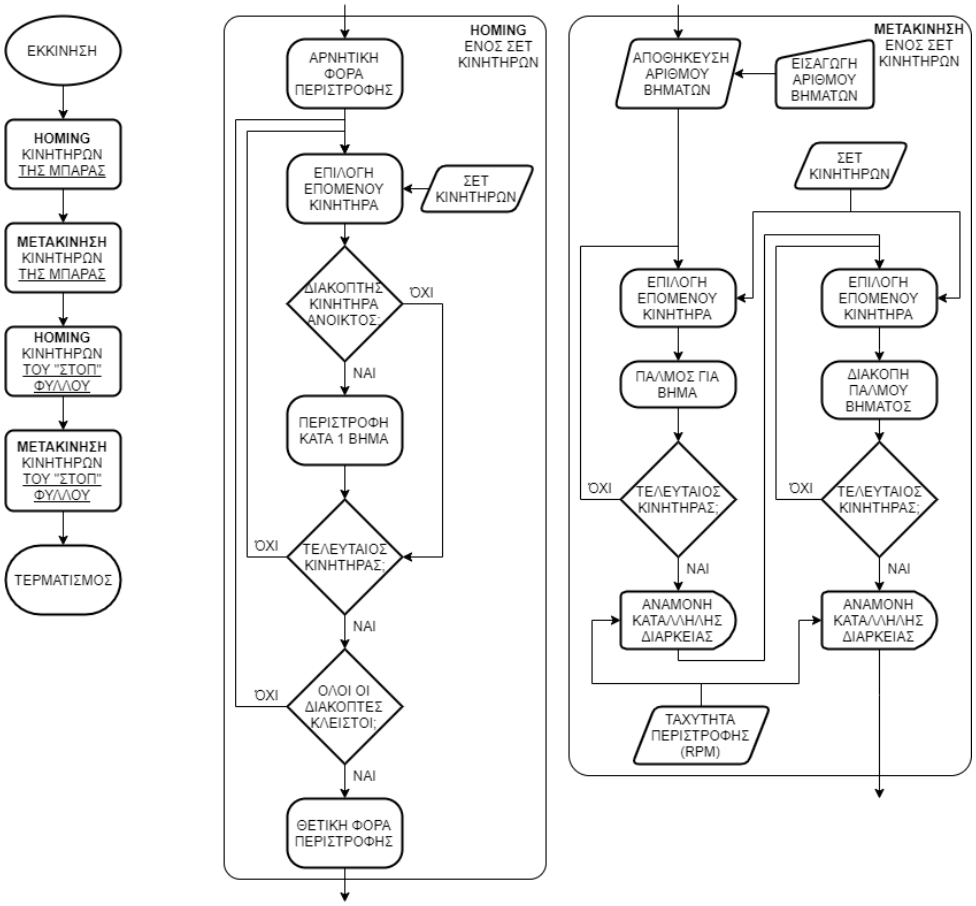



Σχήμα 28: Διάγραμμα συνδεσμολογίας με τον μικρο-ελεγκτή Arduino.

οριακών διακοπτών και αποφασίζεται αν θα συνεχιστεί ή όχι η διαδικασία. Αυτός ο χρόνος ελέγχου της συνθήκης του κώδικα παρεμβάλλεται μεταξύ κάθε βήματος των κινητήρων (όπως φαίνεται και στο σχήμα 29).

Το «στοπ» φύλλου στερεώθηκε στον άξονα γεωμετρίας «D» του κινητήρα με χρήση κάθετης οπής και κοχλία M4. Για τη δοκιμή λειτουργίας του καθώς δεν υπήρχε η επίπεδη επιφάνεια της τράπεζας εντός του πλαισίου αυτό τοποθετήθηκε κοντά σε μία γωνία ώστε να ολισθαίνει πάνω στα προφίλ. Αποδείχθηκε ότι ο κινητήρας δεν μπορεί να περιστρέψει το «στοπ» τοποθετημένο κατά αυτόν τον τρόπο ακόμα και μετά από τοπική λείανσή του. Όμως κατά τη δοκιμή κίνησής του αναποδογυρισμένο ενώ ακουμπά μόνο στην κάτω πλευρά της βάσης του κινητήρα του (βλ. εικόνα 14, κάτω) αυτό κινήθηκε χωρίς πρόβλημα. Ενδεχομένως το μεγάλο βάρος του σε συνδυασμό με μία μη ιδιαίτερα λεία επιφάνεια ολίσθησης να προκαλούν υπερβολική αντίσταση στον κινητήρα (το «στοπ» δεν είναι σφιχτά τοποθετημένο μεταξύ της βάσης και του προφίλ αλλά έχει 0.5mm περιθώριο).

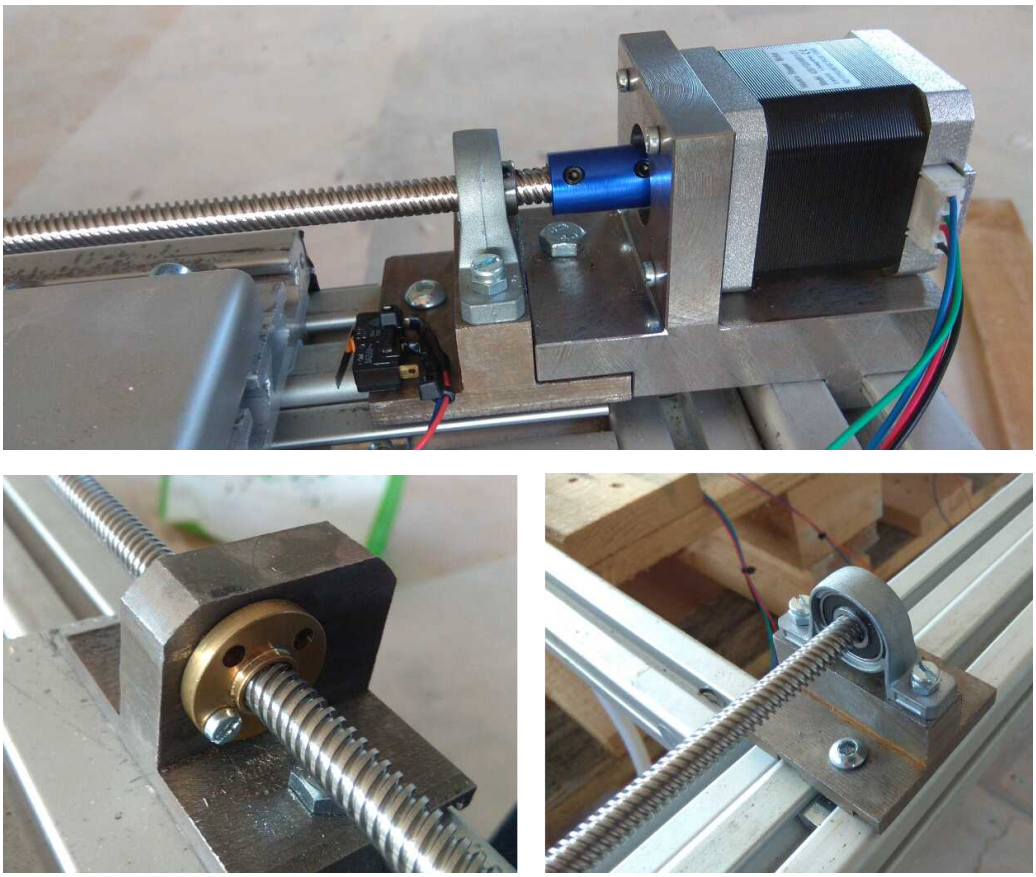
Η αποτελεσματικότητα της κατασκευασμένης λύσης έναντι της χειρωνακτικής μεθόδου διαπιστώθηκε με την σύγκριση που ακολουθεί. Λήφθηκαν υπόψη μόνο οι περιπτώσεις κατακόρυφου και οριζόντιου συλλέκτη για τις οποίες και σχεδιάστηκε η κατασκευασμένη διάταξη. Η χειρωνακτική προετοιμασία της μίας τράπεζας απαιτεί σε κάθε περίπτωση την αφαίρεση των προηγούμενων αντικειμένων από την τράπεζα και προετοιμασία και τοποθέτησή τους στις νέες θέσης. Αυτή η διαδικασία απαιτεί χρόνο από 5 μέχρι 10 λεπτά. Η ρύθμιση της




Σχήμα 29: Διάγραμμα ροής του προγράμματος που χρησιμοποιήθηκε στον μικρο-ελεγκτή Arduino.

αυτόματης διάταξης απαιτεί πρώτα την επαναφορά κάθε κινούμενου μέλους (homing) και στη συνέχεια την μετακίνησή τους στις τελικές θέσεις. Χρησιμοποιήθηκε η υπόθεση ότι στην αρχή της ρύθμισης κάθε κινούμενο μέλος βρίσκεται στη μέση του εύρους κίνησής του, και ότι μετά το homing θα πρέπει να διανύσει όλο το εύρος κίνησής του, καθώς αυτό είναι το χειρότερο δυνατό σενάριο. Συνεπώς, για δεδομένη ταχύτητα περιστροφής 50rpm και υποθέτοντας ότι τα μέλη κινούνται σειριακά ο συνολικός χρόνος ρύθμισης υπολογίζεται ως εξής:

$$
\begin{aligned}
t_{\text{setup}} &= t_{\text{homing}} + t_{\text{moving}} \\
&= \big(t_{\text{h,vert.bar}} + t_{\text{h,horiz.bar}} + t_{\text{h,vert.stops}} + t_{\text{h,horiz.stops}}\big) + \\
&\quad \big(t_{\text{m,vert.bar}} + t_{\text{m,horiz.bar}} + t_{\text{m,vert.stops}} + t_{\text{m,horiz.stops}}\big) \\
&= \frac{500\,\text{mm}/2 + 1000\,\text{mm}/2}{\frac{8\,\text{mm}}{1\text{περιστρ.}}50\,\text{rpm}} + \frac{60^\circ/2 + 60^\circ/2}{\frac{360^\circ}{1\text{περιστρ.}}10\,\text{rpm}} + \\
&\quad \frac{500\,\text{mm} + 1000\,\text{mm}}{\frac{8\,\text{mm}}{1\text{περιστρ.}}50\,\text{rpm}} + \frac{60^\circ + 60^\circ}{\frac{360^\circ}{1\text{περιστρ.}}10\,\text{rpm}} \\
&= 1.892 + 3.783 \\
&= 1'54'' + 3'46'' \\
&= 5'40''
\end{aligned}
$$

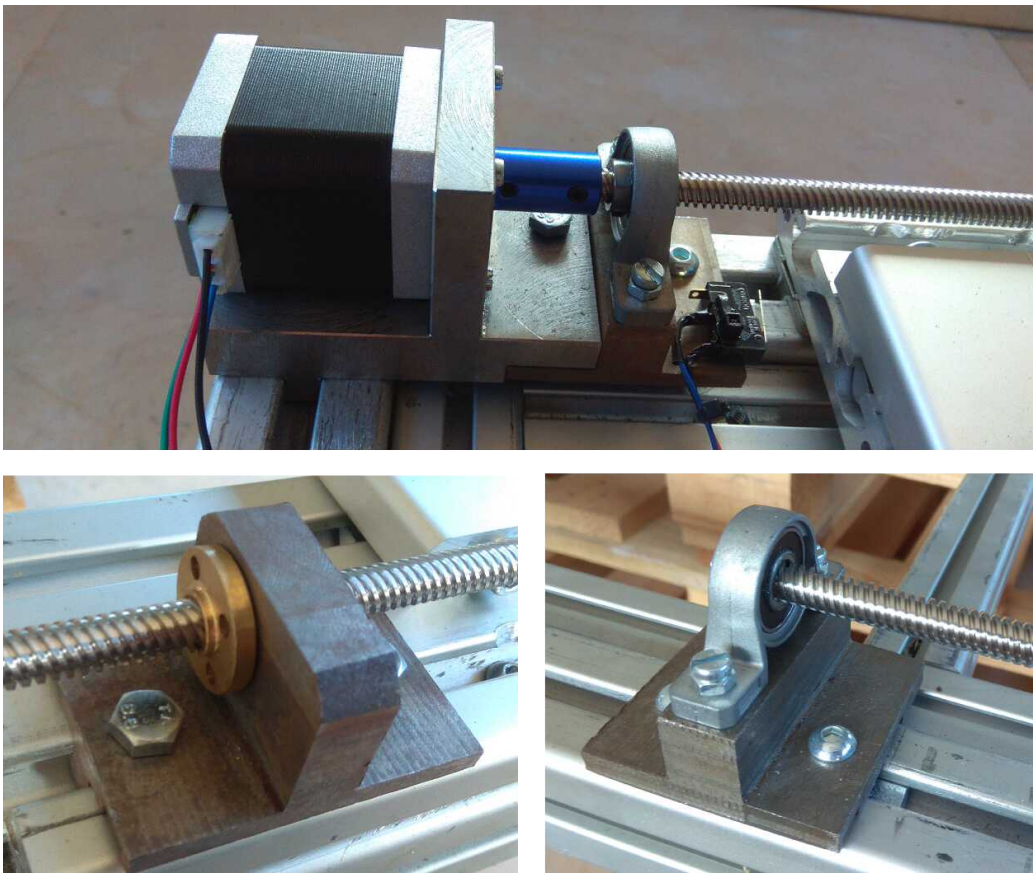


Εικόνα 12: Συναρμολογημένο αριστερό άκρο της μπάρας.

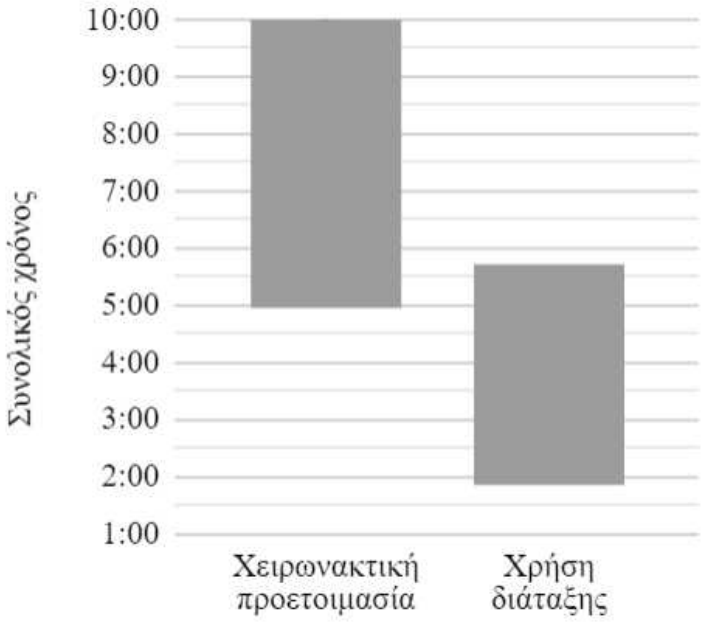
Συνεπώς χρειάζονται κατά μέσο όρο 1′54’ για το homing της διάταξης και στο ακραίο σενάριο όπου χρειάζεται η μεγαλύτερη δυνατή μετακίνηση κάθε μέλους χρειάζονται επιπλέον 3′46’ με συνολικό χρόνο τελικά 5′40’. Η σύγκριση μεταξύ των ευρών τιμών των χρόνων ρύθμισης απεικονίζεται και στο διάγραμμα 30. Παρόλο που δεν υπάρχουν δεδομένα από ακριβείς μετρήσεις για να πραγματοποιηθεί στατιστικό τεστ είναι εμφανές ότι οι χρόνοι ρύθμισης μειώνονται σχεδόν στο μισό. Στην περίπτωση της διάταξης οι ακραίες τιμές (μεγάλες ή μικρές) εμφανίζονται μόνο σε περιπτώσεις αρκετά μεγάλων ή μικρών συλλεκτών ενώ στις υπόλοιπες περιπτώσεις η ρύθμιση γίνεται συστηματικά στον ίδιο χρόνο. Στην περίπτωση της χειρωνακτικής ρύθμισης συνήθως απαιτείται περισσότερος χρόνος είτε όταν χρειάζεται να προστεθούν επιπλέον αντικείμενα στην τράπεζα, όπως κατά την μετάβαση από κατακόρυφο σε οριζόντιο και αντίστροφα, είτε λόγω ανθρώπινου σφάλματος. Αξίζει να σημειωθεί ότι η υπάρχουσα μέθοδος ρύθμισης πραγματοποιείται μόνο από τους έμπειρους χειριστές και όχι από όλους και ακόμα και τότε δεν εξαλείφεται η πιθανότητα σφάλματος. Με την αυτόματα ρυθμιζόμενη διάταξη όχι μόνο εξαλείφεται η πιθανότητα σφάλματος και εξασφαλίζεται η συστηματική ακριβής τοποθέτηση αλλά δημιουργείται και η δυνατότητα ρύθμισης από οποιονδήποτε ρυθμιστή αυξάνοντας έτσι την ευελιξία ως προς την αξιοποίηση του προσωπικού.

## 5.3 Αξιολόγηση αποτελεσμάτων

Λαμβάνοντας υπόψη τα παραπάνω προκύπτει ότι η διάταξη της μπάρας μπορεί να χρησιμοποιηθεί για τη στήριξη πλέγματος σωλήνων μαζί με μία δεύτερη οριζόντια ίδιας λειτουργίας. Εφόσον στο σημείο επαφής των δύο μπαρών χρησιμοποιηθούν ολισθητές από POM οι κινητήρες είναι σε θέση να κινήσουν και τις δύο μπάρες (μία κάθε φορά). Το «στοπ» φύλλου



Εικόνα 13: Συναρμολογημένο δεξί άκρο της μπάρας.




Σχήμα 30: Διάγραμμα μέγιστου και ελάχιστου απαιτούμενου χρόνου ρύθμισης για τις περιπτώσεις της χειρωνακτικής και της αυτόματης προετοιμασίας της τράπεζας.

δεν είναι πλήρως λειτουργικό στην υπάρχουσα κατάσταση αλλά έχει αποδειχθεί ότι υπό βελ-

Εικόνα 14: Συναρμολογημένο «στοπ» φύλλου.

τιωμένες συνθήκες ο κινητήρας είναι σε θέση να το κινήσει με ακρίβεια. Όλα τα στοιχεία στερεώνονται πάνω σε προφίλ της τράπεζας και ο περιορισμός ύψους ικανοποιείται με α-σφάλεια ενώ μειώνονται σημαντικά οι χρόνοι προετοιμασίας και η πιθανότητα ανθρώπινου σφάλματος.

Για τη βελτίωση της μελλοντικής χρήσης της διάταξης παρατίθενται ορισμένες προτάσεις με βάση τις παρατηρήσεις κατά τη δοκιμή. Είναι σημαντικό όλες οι επιφάνειες των προφίλ στις οποίες υπάρχει ολίσθηση με κινούμενο μέρος της διάταξης να είναι καθαρισμένες και πιθανώς λειασμένες ώστε να εξασφαλίζεται όσο το δυνατόν ομαλότερη μετακίνηση των μελών και η μείωση του θορύβου από τα βήματα των κινητήρων. Σε αυτό μπορεί να συμβάλλει και η χρήση micro-stepping για το οποίο είναι ικανοί οι επιλεγμένοι οδηγητές. Επίσης παρόλο που δεν αποτέλεσε σημαντικό πρόβλημα στην δοκιμή η κάμψη που εμφανίζεται στο κέντρο της μπάρας είναι προτιμότερο να αποφευχθεί. Πιθανή λύση είναι η χρήση μικρής ρόδας στο κέντρο της, κυλιόμενη στην επιφάνεια της τράπεζας, ή η χρήση αποστάτη ενώ η τοποθέτηση και της οριζόντιας μπάρας μπορεί να βοηθήσει σε μικρό βαθμό. Αν και ήδη υπάρχει προειδοποίηση και αποφυγή ακραίων τιμών μέσω του προγράμματος υπολογισμού των βημάτων, για να αποφευχθεί η καταπόνηση των κινητήρων σε περίπτωση σφάλματος θα ήταν επιθυμητή η τοποθέτηση οριακών διακοπτών και στις βάσεις στο πέρας της διαδρομής της μπάρας. Επίσης, αν κριθεί απαραίτητο οι ατέρμονες μπορούν εύκολα να αντικατασταθούν με μακρύτερους ίδιας διαμέτρου (⌀8) και βήματος (8mm).

Όσον αφορά την βελτίωση της κίνησης του «στοπ» απαραίτητη είναι η λέπτυνση ή η λείανση του ή με άλλον τρόπο η εξασφάλιση χαμηλής τριβής κατά την περιστροφή του. Εναλλακτικά θα μπορούσε να κατασκευαστεί από ελαφρύτερο υλικό όπως αλουμίνιο ή κάποιο πολυμερές

έτσι ώστε να μειωθεί και η ροπή αδράνειας για την περιστροφή του αλλά και η τριβή με την τράπεζα (λόγω μικρότερου βάρους του). Μία άλλη λύση είναι η χρήση ισχυρότερου κινητήρα η η προσθήκη μειωτήρα κάθετης γωνίας. Καθώς για εξοικονόμηση ύψους ο άξονας του κινητήρα διέρχεται μερικώς στην εσοχή του προφίλ δεν είναι εφικτό να χρησιμοποιηθεί ένας μεγαλύτερης κατηγορίας διαστάσεων (λ.χ. NEMA 23) γιατί τότε θα προεξείχε πέρα από την πλευρά της τράπεζας. Υπάρχουν όμως μοντέλα της ίδιας κατηγορίας διαστάσεων (NEMA 17) μεγαλύτερου μήκους (60mm αντί για 48mm) με περισσότερη ροπή (65N cm αντί για 48N cm. Ένας τέτοιος κινητήρας μπορεί να στερεωθεί στην ίδια βάση με τον προηγούμενο και το συνολικό ύψος μαζί με το ύψος της βάσης οριακά δεν θα υπερβαίνει το όριο ($60\,\mathrm{mm}+17\,\mathrm{mm} = 77\,\mathrm{mm} < 80\,\mathrm{mm}$).

Τέλος, για μεγαλύτερη ευελιξία και δυνατότητα επέκτασης της λειτουργίας της διάταξης προτείνεται η χρήση ισχυρότερου μικρο-ελεγκτή. Στο ενδεχόμενο που θα χρειαστεί να συνδεθούν επιπλέον κινητήρες ή διακόπτες θα χρειαστεί υποχρεωτικά να χρησιμοποιηθεί άλλος μικρο-ελεγκτής ή ένας πολυπλέκτης. Ακόμη, η μεγαλύτερη συχνότητα υπολογισμού μπορεί εξαλείψει το πρόβλημα της καθυστέρησης κατά το homing εξαιτίας των ενδιάμεσων ελέγχων. Με τη χρήση ενός μικρο-ελεγκτή που διαθέτει αποθηκευτικό χώρο και λειτουργικό σύστημα μπορεί ακόμα να επιτευχθεί η απευθείας επικοινωνία μεταξύ του προγράμματος υπολογισμού και του προγράμματος οδήγησης των κινητήρων. Καθώς ο μικρο-ελεγκτής που χρησιμοποιήθηκε δεν διαθέτει αποθηκευτικό χώρο και λειτουργικό σύστημα δεν ήταν δυνατό να εκτελεστεί σε αυτόν πρόγραμμα με γραφικό περιβάλλον και να αποθηκευθούν αρχεία (για αυτό και χρησιμοποιήθηκε η χειρωνακτική εισαγωγή των βημάτων στην σειριακή κονσόλα του).

# Αναφορές


[1] M.E. Merchant. Personal Communication, October 12. 1983; cited in Computerized Manufacturing Automation: Employment, Education, and the Workplace, Washington, D.C., U.S. Congress, Office of Technology Assessment, OTA-CIT-235, April 1984, pp. 60-61.

[2] Computerized Manufacturing Automation: Employment, Education, and the Workplace, Washington, D.C., U.S. Congress, Office of Technology Assessment, OTA-CIT-235, April 1984, p. 96.

[3] B.S. Thompson. "Flexible Fixturing–A Current Frontier in the Evolution of Flexible Manufacturing Cells", ASME Paper No. 84-WA/Prod-16, 1984.

[4] Y. Zheng & W.H. Qian. "A 3-D modular fixture with enhanced localization accuracy and immobilization capability", Int. J. Mach. Tools Manuf., Vol. 48, 2008, pp. 677–687.

[5] E. Hoffman. "Jig and fixture design", 5th ed. Delmar: Cengage Learning, 2004.

[6] H. Wang, Y. Rong, , H. Li, & P. Shaun. "Computer aided fixture design: Recent research and trends", Comput. Des., Vol. 42, 2010, pp. 1085–1094.

[7] H. Hammer, J. Schuster. "A New Approach for Flexible Automation in Boring and Milling Operations", Tages Zeitung fur Metallbearbeitung, Vol. 77, No. 5, 1983, pp. 7-17.

[8] J.R. Dai A.Y.C. Nee, J.Y.H. Fuh, & A. Senthil kumar. "An approach to automating modular fixture design and assembly", Proc. Instn. Mech. Engrs., Vol 211, 1997.

[9] R.C. Brost, K.Y. Goldberg. "A complete algorithm for designing planar fixtures using modular components", IEEE Trans Robots Automat, 1996, Volume 12, No. 1, 1996, pp. 31—46.

[10] Y. Rong, Y. Bai. "Automated generation of fixture configuration design", ASME Trans. Manufacturing Sci Eng, Vol. 119, No. 2, 1997, pp. 208—19.

[11] H.L. Phuah. "Part-Fixture Behaviour Prediction Methodology for Fixture Design Verification", PhD Dissertation, The University of Nottingham, Nottingham UK, 2005.

[12] I. Boyle, Y. Rong, & D.C. Brown. "A review and analysis of current computer-aided fixture design approaches", Robot. Comput. Integr. Manuf., Vol. 27, 2011, pp. 1–12.

[13] Z.M. Bi, & W.J. Zhang. "Flexible fixture design and automation: Review, issues and future direction", Int. J. Prod. Res., Vol. 39, No. 13, 2001, pp. 2867–2894.

# Παράρτημα Α Κώδικας προγράμματος παραγωγής εντολής εργασίας

## main.py

```
from gui_utils import Gui


def main():
    gui = Gui()
    gui.run()


if __name__ == '__main__':
    main()
```

## gui_utils.py

```
import sys
from os import environ, path
from drawing_utils import Drawing
from file_utils import FileManager
import re
# from pprint import pprint

if sys.version_info[0] >= 3:
    import PySimpleGUI as sg
else:
    import PySimpleGUI27 as sg


class Gui:
    def __init__(self):
        LCOL_WIDTH = 21
        FRAME_LCOL_WIDTH = 6
        FRAME2_LCOL_WIDTH = 13
        SETTINGS_LCOL_WIDTH = 8
        SETTINGS2_LCOL_WIDTH = 3
        TB_WID = 7
        self.was_horizontal = False

        HOMEPATH = 'C:' + environ['HOMEPATH']
        if getattr(sys, 'frozen', False):
            CURRENT_DIR = path.dirname(path.realpath(sys.executable
                ))
        elif __file__:
            CURRENT_DIR = path.dirname(path.realpath(__file__))
        self.guessed_name = 'job_order'
        self.default_output_folder = HOMEPATH + '\\Desktop\\'
        self.output_file = (self.default_output_folder +
                            self.guessed_name + '.pdf')
        self.OLDS_DIR = 'old'
        self.SETTINGS_FILE = CURRENT_DIR + '\\settings.txt'
        self.SETTINGS_FILE_EXISTS = path.isfile(self.SETTINGS_FILE)

        # machine default settings
        self.default_settings = {
            'vel': 210,
            'acc': 350,
            'pres': 1.6,
            'Z1': 5,  # Z for riser diam. 8 or smaller
            'Z2': 4,  # Z for riser larger than diam. 8
            'P1': 6900,     # -------------------
            'pulse1': 0.3,  # -  for selective  -
            'f1': 155,      # -------------------
            'P2': 7000,     # -------------------
            'pulse2': 0.4,  # - for selective04 -
            'f2': 125,      # -------------------
            'P3': 6700,     # -------------------
            'pulse3': 0.3,  # -     for TSS     -
            'f3': 160,      # -------------------
            'P4': 6700,     # -------------------
            'pulse4': 0.3,  # -   for painted   -
            'f4': 160,      # -------------------
            'P5': 8200,     # -------------------
            'pulse5': 0.4,  # -   for copper    -
            'f5': 109       # -------------------
        }

        # machine settings
        self.settings = self.default_settings.copy()

        # visual options
        sg.SetOptions(
            icon=None,
            button_color=(None, None),
            element_size=(None, None),
            margins=(None, None),
            element_padding=(None, None),
            auto_size_text=None,
            auto_size_buttons=None,
            font=None,
            border_width=None,
            slider_border_width=None,
            slider_relief=None,
            slider_orientation=None,
            autoclose_time=None,
            message_box_line_width=None,
            progress_meter_border_depth=None,
            progress_meter_style=None,
            progress_meter_relief=None,
            progress_meter_color=None,
            progress_meter_size=None,
            text_justification=None,
            background_color=None,
            element_background_color=None,
            text_element_background_color=None,
            input_elements_background_color=None,
            scrollbar_color=None, text_color=None,
            debug_win_size=(None, None),
            window_location=(None, None)
        )

        # define main window layout
        self.layout = [
            # absobrber info
            [sg.Text('Όνομα/Κωδικός/πληροφορίες + '\n' + απορροφητή,
             size=(LCOL_WIDTH, 2),
             auto_size_text=False,
             justification='right'),
             sg.Multiline('', do_not_clear=True, size=(25, 3),
                          enable_events=True, key='absorber_info')
                            ],

            # absorber drawing number
            [sg.Text(Αριθμός Σχεδίου,
             size=(LCOL_WIDTH, 1),
```

```
            auto_size_text=False,
            justification='right'),
            sg.InputText(do_not_clear=True, size=(25, 1),
            enable_events=True, key='drawing_number')],

        # absorber type
        [sg.Text(Τύπος απορροφητή, size=(LCOL_WIDTH, 2),
            auto_size_text=False,
            justification='right'),
            sg.Frame(layout=[
                [sg.Radio(Κατακόρυφος, 'absorber_type_radio',
                    default=True,
                    enable_events=True,
                    key='is_vertical')],
                [sg.Radio(Οριζόντιος, 'absorber_type_radio',
                    enable_events=True,
                    key='is_horizontal')],
                [sg.Radio(Με λωρίδες (φινάκια), 'absorber_type_radio',
                    enable_events=True,
                    key='is_strips')],
                [sg.Radio(Με μαίανδρο, 'absorber_type_radio',
                    enable_events=True,
                    key='is_meander')]
            ],
                title='',
                relief=sg.RELIEF_FLAT
            )],

        # panel/strips section
        [sg.Frame(Χαρακτηριστικά φύλλου/λωρίδων  + '(mm)', layout=[
            # panel/strips material
            [sg.Text(Τλικό, size=(FRAME_LCOL_WIDTH, 2),
                    auto_size_text=False,
                    justification='right'),
             sg.Frame('', [
                [sg.Radio(Επιλεκτικό, 'panel_mat_radio',
                    default=True, key='is_selective')],
                [sg.Radio(Επιλεκτικό 0.4, 'panel_mat_radio',
                    key='is_selective04')],
                [sg.Radio('TSS', 'panel_mat_radio',
                    key='is_tss')],
                [sg.Radio(Βαμμένο, 'panel_mat_radio',
                    key='is_painted')],
                [sg.Radio(Χαλκός, 'panel_mat_radio',
                    key='is_copper')]
             ],
                relief=sg.RELIEF_FLAT
             )],

            # panel/strips dimensions
            [sg.Text(Μήκος, size=(FRAME_LCOL_WIDTH, 1),
             auto_size_text=False,
             justification='right',
             key='panel_length_label'),
             sg.InputText(do_not_clear=True, size=(TB_WID, 1),
             key='panel_length')],

            [sg.Text(Πλάτος, size=(FRAME_LCOL_WIDTH, 1),
             auto_size_text=False,
             justification='right',
             key='panel_width_label'),
             sg.InputText(do_not_clear=True, size=(TB_WID, 1),
             key='panel_width')],
```



```
            # [sg.Text(Αριθμός, size=(FRAME_LCOL_WIDTH, 1),
            #  auto_size_text=False,
            #  justification='right', text_color='gray',
            #  key='strip_count_label'),
            #  sg.InputText(do_not_clear=True, size=(TB_WID, 1)
                ,
            #  key='strip_count', enable_events=True, disabled=
                True)],

            [sg.Text('', size=(FRAME_LCOL_WIDTH, 1)),
             sg.Checkbox(Οπές στις γωνίες,
             default=False,
             key='panel_holes')]
        ]),

        # grid section
        sg.Frame(Χαρακτηριστικά  + 'grid' +  σωληνών  + '(mm)',
            layout=[
            # header diameter
            [sg.Text(Φ + ' header', size=(FRAME2_LCOL_WIDTH, 1)
                ,
             auto_size_text=False,
             justification='right'),
             sg.InputText('22', do_not_clear=True, size=(TB_WID
                , 1),
             key='header_diameter')],

            # riser diameter
            [sg.Text(Φ + ' riser', size=(FRAME2_LCOL_WIDTH, 1),
             auto_size_text=False,
             justification='right'),
             sg.InputText('8', do_not_clear=True, size=(TB_WID,
                1),
             key='riser_diameter')],

            # head to head distance
            [sg.Text(Κέντρο-κέντρο, size=(FRAME2_LCOL_WIDTH, 1),
             auto_size_text=False,
             justification='right'),
             sg.InputText(do_not_clear=True, size=(TB_WID, 1),
             key='head_to_head')],

            # free header exit length
            [sg.Text(Μήκος ελεύθερης + '\n' + εξόδου + 'header',
             size=(FRAME2_LCOL_WIDTH, 2),
             auto_size_text=False,
             justification='right'),
             sg.InputText(do_not_clear=True, size=(TB_WID, 1),
             key='header_exit_length')],

            # header total length
            [sg.Text(Μήκος  + 'header\n' + με κλειστή έξοδο,
             size=(FRAME2_LCOL_WIDTH, 2),
             auto_size_text=False,
             justification='right', text_color='gray',
             key='header_length_label'),
             sg.InputText(do_not_clear=True, size=(TB_WID, 1),
             key='header_length', disabled=True)],

            # closed exits
            [sg.Text('', size=(int(FRAME2_LCOL_WIDTH/2-2), 1)),
             sg.Frame(Κλειστές έξοδοι, [
                [sg.Checkbox('', default=False,
                 enable_events=True,
```

```
                    key='up_left_exit_closed'),
                    sg.Checkbox('', default=False,
                    enable_events=True,
                    key='up_right_exit_closed')],
                   [sg.Checkbox('', default=False,
                    enable_events=True,
                    key='down_left_exit_closed'),
                    sg.Checkbox('', default=False,
                    enable_events=True,
                    key='down_right_exit_closed')]
                ])],

            # distance of 1st riser from edge
            [sg.Text(Απόσταση α + '\' riser\n'+ από άκρη φύλλου,
             size=(FRAME2_LCOL_WIDTH, 2),
             auto_size_text=False,
             justification='right'),
             sg.InputText(do_not_clear=True, size=(TB_WID, 1),
             key='riser_edge_distance')],

            # riser step
            [sg.Text(Βήμα
                + 'riser', size=(FRAME2_LCOL_WIDTH, 1),
             auto_size_text=False,
             justification='right'),
             sg.InputText(do_not_clear=True, size=(TB_WID, 1),
             key='riser_step')],

            # riser count
            [sg.Text(Αριθμός
               +'riser', size=(FRAME2_LCOL_WIDTH,1),
             auto_size_text=False,
             justification='right', key='riser_count_label'),
             sg.InputText(do_not_clear=True, size=(TB_WID, 1),
             key='riser_count', enable_events=True)],

            # riser count
            [sg.Text(Πλάτος
               μαιάνδρου, size=(FRAME2_LCOL_WIDTH, 1),
             auto_size_text=False,
             justification='right', text_color='gray',
             key='meander_width_label'),
             sg.InputText(do_not_clear=True, size=(TB_WID, 1),
             key='meander_width', disabled=True)],
         ])],

        # error box
        [sg.Text('', text_color='red', size=(60, 2),
                 key='error_box')],

        # buttons
        [sg.Text(' ' * 35),
         sg.Button(Δημιουργία, size=(10, 1), key='save_button'),
         sg.Button('Εξοδος, size=(10, 1), key='exit_button'),
         sg.Text(' ' * 12),
         sg.Button(Ρυθμίσεις + '\n' + μηχανής, size=(8, 2),
                   key='settings_button')]
    ]

    # define machine settings window layout
    self.settings_layout = [
        [sg.Frame('', layout=[
            [sg.Text('Vel', size=(SETTINGS2_LCOL_WIDTH, 1),
             auto_size_text=False,
```



```
                justification='right'),
                sg.InputText('', do_not_clear=True, size=(6, 1),
                enable_events=True,
                key='vel')],
               [sg.Text('Acc', size=(SETTINGS2_LCOL_WIDTH, 1),
                auto_size_text=False,
                justification='right'),
                sg.InputText('', do_not_clear=True, size=(6, 1),
                enable_events=True,
                key='acc')],
               [sg.Text('Pres', size=(SETTINGS2_LCOL_WIDTH, 1),
                auto_size_text=False,
                justification='right'),
                sg.InputText('', do_not_clear=True, size=(6, 1),
                enable_events=True,
                key='pres')],
               [sg.Text('Z', size=(SETTINGS2_LCOL_WIDTH, 1),
                auto_size_text=False,
                justification='right'),
                sg.InputText('', do_not_clear=True, size=(6, 1),
                enable_events=True,
                key='Z')]
            ]),
            sg.Frame('', layout=[
               [sg.Text('Power (W)', size=(SETTINGS_LCOL_WIDTH, 1)
                   ,
                auto_size_text=False,
                justification='right'),
                sg.InputText('', do_not_clear=True, size=(6, 1),
                enable_events=True,
                key='P')],
               [sg.Text('Pulse (ms)', size=(SETTINGS_LCOL_WIDTH,
                  1),
                auto_size_text=False,
                justification='right'),
                sg.InputText('', do_not_clear=True, size=(6, 1),
                enable_events=True,
                key='pulse')],
               [sg.Text('Freq (Hz)', size=(SETTINGS_LCOL_WIDTH, 1)
                  ,
                auto_size_text=False,
                justification='right'),
                sg.InputText('', do_not_clear=True, size=(6, 1),
                enable_events=True,
                key='f')]
            ])],

        # error box
        [sg.Text('', text_color='red', size=(35, 2),
                 key='settings_error_box')],

        # buttons
        [sg.Button('OK', size=(10, 1),
                   key='OK_button'),
         sg.Button(Αποθήκευση, size=(10, 1),
                   key='export_button'),
         sg.Button(Επαναφορά, size=(10, 1),
                   key='reset_button')]
    ]

    # drawing instance
    self.drawing = Drawing()

    # file manager instance
```

```
        self.file_manager = FileManager()

    def set_to_strips(self):
        # (self.window.Element('strip_count_label')
        #  .Update(text_color='black'))
        # self.window.Element('strip_count').Update(disabled=False)
        self.window.Element('panel_holes').Update(disabled=True)
        self.window.Element('is_selective').Update(value=True)
        self.window.Element('is_tss').Update(disabled=True)
        self.window.Element('is_painted').Update(disabled=True)
        self.window.Element('is_copper').Update(disabled=True)

    def set_to_panel(self):
        # (self.window.Element('strip_count_label')
        #  .Update(text_color='gray'))
        # self.window.Element('strip_count').Update(disabled=True)
        self.window.Element('panel_holes').Update(disabled=False)
        self.window.Element('is_tss').Update(disabled=False)
        self.window.Element('is_painted').Update(disabled=False)
        self.window.Element('is_copper').Update(disabled=False)

    def set_to_meander(self):
        (self.window.Element('meander_width_label')
         .Update(text_color='black'))
        self.window.Element('meander_width').Update(disabled=False)

    def unset_to_meander(self):
        (self.window.Element('meander_width_label')
         .Update(text_color='gray'))
        self.window.Element('meander_width').Update(disabled=True)

    def reset_grid(self, is_horizontal):
        # if needed clear grid
        if is_horizontal != self.was_horizontal:
            self.clear_grid()
        self.was_horizontal = is_horizontal

    def clear_grid(self):
        self.window.Element('header_diameter').Update('22')
        self.window.Element('riser_diameter').Update('8')
        self.window.Element('head_to_head').Update('')
        self.window.Element('header_exit_length').Update('')
        self.window.Element('riser_edge_distance').Update('')
        self.window.Element('riser_step').Update('')
        (self.window.Element('up_left_exit_closed')
         .Update(False, disabled=False))
        (self.window.Element('up_right_exit_closed')
         .Update(False, disabled=False))
        (self.window.Element('down_left_exit_closed')
         .Update(False, disabled=False))
        (self.window.Element('down_right_exit_closed')
         .Update(False, disabled=False))

        (self.window.Element('header_length_label')
         .Update(text_color='gray'))
        (self.window.Element('header_length')
         .Update(disabled=True))

    def set_exits(self, values):
        is_horizontal = values['is_horizontal']
        if not is_horizontal:
            # allow only one of up exits closed
            if values['up_left_exit_closed']:
                (self.window.Element('up_right_exit_closed').
```



```
                 Update(disabled=True))
            else:
                (self.window.Element('up_right_exit_closed').
                 Update(disabled=False))

            if values['up_right_exit_closed']:
                (self.window.Element('up_left_exit_closed').
                 Update(disabled=True))
            else:
                (self.window.Element('up_left_exit_closed').
                 Update(disabled=False))

            # allow only one of down exits closed
            if values['down_left_exit_closed']:
                (self.window.Element('down_right_exit_closed').
                 Update(disabled=True))
            else:
                (self.window.Element('down_right_exit_closed').
                 Update(disabled=False))

            if values['down_right_exit_closed']:
                (self.window.Element('down_left_exit_closed').
                 Update(disabled=True))
            else:
                (self.window.Element('down_left_exit_closed').
                 Update(disabled=False))
        else:
            # allow only one of left exits closed
            if values['up_left_exit_closed']:
                (self.window.Element('down_left_exit_closed').
                 Update(disabled=True))
            else:
                (self.window.Element('down_left_exit_closed').
                 Update(disabled=False))

            if values['down_left_exit_closed']:
                (self.window.Element('up_left_exit_closed').
                 Update(disabled=True))
            else:
                (self.window.Element('up_left_exit_closed').
                 Update(disabled=False))

            # allow only one of right exits closed
            if values['up_right_exit_closed']:
                (self.window.Element('down_right_exit_closed').
                 Update(disabled=True))
            else:
                (self.window.Element('down_right_exit_closed').
                 Update(disabled=False))

            if values['down_right_exit_closed']:
                (self.window.Element('up_right_exit_closed').
                 Update(disabled=True))
            else:
                (self.window.Element('up_right_exit_closed').
                 Update(disabled=False))

        # disable/enable header length text box
        if True in (values['up_left_exit_closed'],
                    values['up_right_exit_closed'],
                    values['down_left_exit_closed'],
                    values['down_right_exit_closed']):
            (self.window.Element('header_length_label')
             .Update(text_color='black'))
```

```
            (self.window.Element('header_length')
             .Update(disabled=False))
        else:
            (self.window.Element('header_length_label')
             .Update(text_color='gray'))
            (self.window.Element('header_length')
             .Update(disabled=True))

    def is_float(self, string):
        try:
            if float(string) > 0.0:
                return True
            else:
                return False
        except ValueError:
            return False

    def check_for_errors(self, values):
        # check for positive floats
        try:
            for key in values.keys():
                # skip absorber info
                if key in ('absorber_info', 'drawing_number'):
                    continue

                # skip disabled values for each casa
                if not values['is_strips']:
                    if key == 'strip_count':
                        continue
                if not values['is_meander']:
                    if key == 'meander_width':
                        continue
                if True not in (values['up_left_exit_closed'],
                                values['up_right_exit_closed'],
                                values['down_left_exit_closed'],
                                values['down_right_exit_closed']):
                    if key == 'header_length':
                        continue

                if type(values[key]) == str:
                    if not self.is_float(values[key]):
                        return Σφάλμα: επιτρέπονται μόνο θετικοί αριθμοί
        except AttributeError:
            return Σφαλμα: μη έγκυρη τιμή

        # check head-to-head+header diameter and panel width or
           length
        if values['is_horizontal']:
            if (values['head_to_head'] + values['header_diameter']
               >
               values['panel_width']):
                return (Σφάλμα: το πλάτος του φύλλου δεν καλύπτει
                        πλήρως τους headers)
        else:
            if (values['head_to_head'] + values['header_diameter']
               >
               values['panel_length']):
                return (Σφάλμα: το μήκος του φύλλου δεν καλύπτει
                        πλήρως τους headers)

        # check tube diameters
        if values['riser_diameter'] > values['header_diameter']:
            return (Σφαλμα: η διάμετρος των  + 'riser' +
                    δεν μπορεί να είναι μεγαλύτερη από αυτή των  +
```



```
                    'header')

        # check riser step and riser diameter
        if values['riser_step'] < values['riser_diameter']:
            return (Σφαλμα: το βήμα των  + 'riser' +  δεν μπορεί  +
                    να είναι μικρότερο από τη διάμετρό τους)

        # check riser step and panel length or width
        if values['is_horizontal'] or values['is_meander']:
            if values['riser_step'] > values['panel_length']:
                return (Σφαλμα: το βήμα των  + 'riser' +
                        δεν μπορεί να είναι μεγαλύτερο από το  +
                        μήκος του  + '\n' + φύλλου (σε
                        οριζόντιο απορροφητή ή μαίανδρο))
        else:
            if values['riser_step'] > values['panel_width']:
                return (Σφαλμα: το βήμα των  + 'riser' +
                        δεν μπορεί να είναι μεγαλύτερο από το  +
                        συνολκό πλάτος του + '\n' + φύλλου)

        # check riser edge distance and panel width or length
        if values['is_horizontal']:
            if values['riser_edge_distance'] > values['panel_length
               ']:
                return (Σφαλμα: η απόσταση α + '\'' +
                        από την άκρη του φύλλου
                        δεν μπορεί να είναι μεγαλύτερη  + '\n' +
                        από το μήκος του φύλλου (σε οριζόντιο  +
                        απορροφητή))
        else:
            if values['riser_edge_distance'] > values['panel_width'
               ]:
                return (Σφαλμα: η απόσταση α + '\'' +
                        από την άκρη του φύλλου
                        δεν μπορεί να είναι μεγαλύτερη  + '\n' +
                        από το συνολικό πλάτος των φύλλων)

        # check header length and free exit length
        if True in (values['up_left_exit_closed'],
                    values['up_right_exit_closed'],
                    values['down_left_exit_closed'],
                    values['down_right_exit_closed']):
            if values['header_length'] < values['header_exit_length
               ']:
                return (Σφαλμα: το μήκος της ελεύθερης εξόδου δεν
                        μπορεί να είναι μεγαλύτερο από το  + '\n' +
                        μήκος του  + 'header')

        # check for floating risers
        if values['is_horizontal'] or values['is_meander']:
            if ((values['riser_count']-1)*values['riser_step'] +
               values['riser_edge_distance'] > values['panel_length
                  ']):
                return (Σφαλμα: δεν χωράει τόσο μεγάλος αριθμός
                        riser στο μήκος του φύλλου)
        else:
            if ((values['riser_count']-1)*values['riser_step'] +
               values['riser_edge_distance'] > values['panel_width'
                  ]):
                return (Σφαλμα: δεν χωράει τόσο μεγάλος αριθμός
                        riser στο πλατος του φύλλου)

        # check for meander width and panel width
        if values['is_meander']:
```

```
601             if (values['riser_edge_distance'] + values['
                    meander_width'] >
602                 values['panel_width']):
603                  return (Σφάλμα:  ο μαίανδρος  + 'riser' +
604                           είναι μεγαλύτερος σε πλάτος από το φύλλο)
605
606         return ''
607
608     def check_settings_for_errors(self, values):
609         # check for positive floats
610         try:
611             for key in values.keys():
612                 if type(values[key]) == str:
613                     if not self.is_float(values[key]):
614                         return Σφάλμα:  επιτρέπονται μόνο θετικοί αριθμοί
615         except AttributeError:
616             return Σφαλμα:  μη έγκυρη τιμή
617
618         return ''
619
620     def show_error(self, error):
621         self.window.Element('error_box').Update(error)
622
623     def show_settings_error(self, error):
624         self.settings_window.Element('settings_error_box').Update(
                error)
625
626     def export_settings_to_file(self):
627         with open(self.SETTINGS_FILE, 'w', encoding='utf-8') as
                the_file:
628             for key in self.settings.keys():
629                 # write separators
630                 if key == 'Z1':
631                     the_file.write(
632                         ('\n# Z' +  για  + 'riser' +
633                              Φ8 ή μικρότερους + '\n')
634                     )
635                 elif key == 'Z2':
636                     the_file.write(
637                         ('\n# Z' +  για  + 'riser' +
638                             μεγαλύτερους από Φ8 + '\n')
639                     )
640                 elif key == 'P1':
641                     the_file.write(
642                         '\n#'+ ρυθμίσεις για επιλεκτικά φύλλα+'\n'
643                     )
644                 elif key == 'P2':
645                     the_file.write(
646                         ('\n#' +  ρυθμίσεις για επιλεκτικά φύλλα με  +
647                              '0.4\n')
648                     )
649                 elif key == 'P3':
650                     the_file.write(
651                         ('\n#' +  ρυθμίσεις για +'TSS'+ φύλλα + '\n'
652                     )
653                 elif key == 'P4':
654                     the_file.write(
655                         '\n#' +  ρυθμίσεις για βαμμένα φύλλα + '\n'
656                     )
657                 elif key == 'P5':
658                     the_file.write(
659                         '\n#' +  ρυθμίσεις για φύλλα από χαλκό + '\n'
660                     )
661                 the_file.write(key+'=' + str(self.settings[key]) +
```



```
662                  '\n')
663     def import_settings_from_file(self):
664         # reset imported settings to defaults
665         imported_settings = self.default_settings.copy()
666
667         # if custom settings file is present in current directory
                then
668         # import these settings
669         if self.SETTINGS_FILE_EXISTS:
670             with open(self.SETTINGS_FILE, encoding='utf-8') as
                    the_file:
671                 for line in the_file:
672                     # skip comment lines
673                     if line.startswith('#'):
674                         continue
675                     # store appropriate
676                     for key in self.settings.keys():
677                         if line.startswith(key + '='):
678                             imported_settings[key] = (line.strip('\
                                    n')
679                                                       .split('=')
                                                          [1])
680
681         return imported_settings
682
683     def fill_settings_window(self, main_values):
684         # print loaded settings file
685         if self.SETTINGS_FILE_EXISTS:
686             self.show_settings_error(
687                 (Οι αρχικές ρυθμίσεις φορτώθηκαν από το αρχείο:+'\n' +
688                  self.SETTINGS_FILE)
689             )
690
691         for item in ('vel', 'acc', 'pres'):
692             self.settings_window.Element(item).Update(self.settings
                    [item])
693
694         # set Z depending on riser diameter
695         if main_values['riser_diameter'] == '':
696             main_values['riser_diameter'] = 0
697         if main_values['riser_diameter'] <= 8:
698             self.settings_window.Element('Z').Update(self.settings[
                    'Z1'])
699         else:
700             self.settings_window.Element('Z').Update(self.settings[
                    'Z2'])
701
702         # set laser parameters according to panel material
703         if main_values['is_selective']:
704             for item in ('P', 'pulse', 'f'):
705                 (self.settings_window.Element(item)
706                  .Update(self.settings[item + '1']))
707         elif main_values['is_selective04']:
708             for item in ('P', 'pulse', 'f'):
709                 (self.settings_window.Element(item)
710                  .Update(self.settings[item + '2']))
711         elif main_values['is_tss']:
712             for item in ('P', 'pulse', 'f'):
713                 (self.settings_window.Element(item)
714                  .Update(self.settings[item + '3']))
715         elif main_values['is_painted']:
716             for item in ('P', 'pulse', 'f'):
717                 (self.settings_window.Element(item)
```

```
                .Update(self.settings[item + '4']))
        elif main_values['is_copper']:
            for item in ('P', 'pulse', 'f'):
                (self.settings_window.Element(item)
                 .Update(self.settings[item + '5']))

    def run_settings(self, main_values):
        # draw settings window
        self.settings_window = (sg.Window(Ρυθμίσεις Μηχανής,
                                  keep_on_top=True)
                                  .Layout(self.settings_layout))
        self.settings_window.Finalize()

        self.fill_settings_window(main_values)

        # settings window event Loop
        while True:
            # get the event and current values
            event, values = self.settings_window.Read()

            # user closes the window
            if event in (None, 'OK_button'):
                break

            # format values
            for key, val in values.items():
                if self.is_float(val):
                    values[key] = float(val)

            # check for errors in any event except 'reset', 'X', or
                'OK'
            if event not in (None, 'OK_button', 'reset_button'):
                self.show_settings_error('')
                error_found = self.check_settings_for_errors(values
                    )
                if error_found != '':
                    self.show_settings_error(error_found)
                    continue

            # user presses 'export settings file' button
            if event == 'export_button':
                self.export_settings_to_file()

                self.show_settings_error(
                    (Οι ρυθμίσεις αποθηκεύτηκαν στο αρχείο: + '\n' +
                     self.SETTINGS_FILE)
                )

            # user presses 'reset' button
            if event == 'reset_button':
                self.settings = self.default_settings.copy()
                self.fill_settings_window(main_values)

                loaded_from = ''
                if self.SETTINGS_FILE_EXISTS:
                    loaded_from =  από: + '\n' + self.SETTINGS_FILE

                self.show_settings_error(
                    (Οι ρυθμήσεις επαναφέρθηκαν στις αρχικές +
                     loaded_from)
                )

            # if user modifies a value then store it
            if event in ('vel', 'acc', 'pres'):
```



```
                self.settings[event] = values[event]

            if event == 'Z':
                if main_values['riser_diameter'] <= 8:
                    self.settings['Z1'] = values['Z']
                else:
                    self.settings['Z2'] = values['Z']

            if event in ('P', 'pulse', 'f'):
                if main_values['is_selective']:
                    self.settings[event + '1'] = values[event]
                elif main_values['is_selective04']:
                    self.settings[event + '2'] = values[event]
                elif main_values['is_painted']:
                    self.settings[event + '3'] = values[event]
                elif main_values['is_copper']:
                    self.settings[event + '4'] = values[event]

            # DEBUG
            # print('\nsettings event: ')
            # pprint(event)
            # print('\nsettings values: ')
            # pprint(values)

        self.settings_window.Close()

    def run(self):
        # draw main window
        self.window = (sg.Window(Δημιουργία Εντολής Εργασίας)
                       .Layout(self.layout))

        # initialize machine settings
        self.default_settings = self.import_settings_from_file()
        self.settings = self.default_settings.copy()

        # main window vent Loop
        while True:
            # get the event and current values
            event, values = self.window.Read()

            # user closes the window
            if event in (None, 'exit_button'):
                break

            # format values
            for key, val in values.items():
                if key in ('absorber_info', 'drawing_number'):
                    continue
                if self.is_float(val):
                    if key in ('strip_count', 'riser_count'):
                        values[key] = int(float(val))
                    elif 'closed' in key:
                        values[key] = bool(int(val))
                    else:
                        values[key] = float(val)

            # user presses 'save' button
            if event == 'save_button':
                # print errors (if any)
                self.show_error('')
                error_found = self.check_for_errors(values)
                if error_found != '':
                    self.show_error(error_found)
                    continue
```

```
844
845                 # disable buttons
846                 self.window.Element('save_button').Update(disabled=
                        True)
847                 self.window.Element('exit_button').Update(disabled=
                        True)
848                 (self.window.Element('settings_button')
849                  .Update(disabled=True))
850
851                 # prompt user to supply file name
852                 self.output_file = sg.PopupGetFile(
853                     title=Αποθήκευση αρχείου,
854                     message=(Παρακαλώ εισάγετε την τοποθεσία του
855                              παραγόμενου αρχείου),
856                     save_as=True,
857                     default_path=(self.default_output_folder +
858                                   self.guessed_name + '.pdf'),
859                     default_extension='pdf',
860                     file_types=((PDF, *.pdf),)
861                 )
862
863                 # if a name is defined then proceed to save
864                 if self.output_file:
865                     # ensure correct file extension
866                     if not self.output_file.endswith('.pdf'):
867                         self.output_file += '.pdf'
868
869                     # clean up save directory
870                     files_were_moved = self.file_manager.
                            prepare_dir(
871                          values['drawing_number'],
872                          path.dirname(self.output_file), self.
                                OLDS_DIR)
873
874                     if files_were_moved:
875                         moved_files_message = ''
876
877                     # create job order PDF
878                     self.drawing = Drawing()
879                     self.drawing.make_job_order(self.output_file,
                            values,
880                                                 self.settings)
881
882                     self.show_error((Το αρχείο αποθηκεύτηκε στην
883                                      τοποθεσία: + '\n' +
884                                      self.output_file))
885
886                 # re-enable buttons
887                 self.window.Element('save_button').Update(disabled=
                        False)
888                 self.window.Element('exit_button').Update(disabled=
                        False)
889                 (self.window.Element('settings_button')
890                  .Update(disabled=False))
891
892             # user presses 'machine settings' button
893             if event == 'settings_button':
894                 # print errors (if any)
895                 # self.show_error('')
896                 # error_found = self.check_for_errors(values)
897                 # if error_found != '':
898                 #     self.show_error(error_found)
899                 #     continue
900
```



```
901                 # disable buttons
902                 self.window.Element('save_button').Update(disabled=
                        True)
903                 self.window.Element('exit_button').Update(disabled=
                        True)
904                 (self.window.Element('settings_button')
905                  .Update(disabled=True))
906
907                 # open machine settings window
908                 self.run_settings(values)
909
910                 # re-enable buttons
911                 self.window.Element('save_button').Update(disabled=
                        False)
912                 self.window.Element('exit_button').Update(disabled=
                        False)
913                 (self.window.Element('settings_button')
914                  .Update(disabled=False))
915
916             # update panel and grid on product type change
917             if event in ('is_vertical', 'is_horizontal', 'is_strips
                    ',
918                          'is_meander'):
919                 self.reset_grid(values['is_horizontal'])
920
921                 if event == 'is_meander':
922                     self.set_to_meander()
923                 else:
924                     self.unset_to_meander()
925
926                 if event == 'is_strips':
927                     self.set_to_strips()
928                 else:
929                     self.set_to_panel()
930
931             # set exits
932             if event in ('up_left_exit_closed',
933                          'up_right_exit_closed',
934                          'down_left_exit_closed',
935                          'down_right_exit_closed'):
936                 self.set_exits(values)
937
938             # force integers in 'strip count' field
939             if event in ('strip_count', 'riser_count'):
940                 if isinstance(values[event], str):
941                     cur_value = values[event].replace(',', '.')
942                     if self.is_float(cur_value):
943                         self.window.Element(event).Update(
944                             (str(int(float(cur_value)))))
945                 else:
946                     self.window.Element(event).Update(int(values[
                            event]))
947
948             # set possible file name and set panel material
949             if event in ('absorber_info', 'drawing_number'):
950                 # format name
951                 guessed_name = (values['absorber_info']
952                                 .split('\n')[0]
953                                 .strip(',|.| |\t'))
954
955                 # format guessed name and drawing number
956                 for char in ['/', '\\', '|', '*']:
957                     guessed_name = guessed_name.replace(char, '-')
958                     values['drawing_number'] = (values['
```

```
                    drawing_number']
                                        .replace(char, '-')
                                         )

                for char in ['\t', ':', '?', '<', '>']:
                    guessed_name = guessed_name.replace(char, '')
                    values['drawing_number'] = (values['
                        drawing_number']
                                        .replace(char, ''))

                self.window.Element('drawing_number').Update(
                    values['drawing_number'])

                # append drawing number, if any
                if values['drawing_number'] != '':
                    guessed_name = (values['drawing_number'] + '-'
                        +
                                    guessed_name)

                if guessed_name != '':
                    self.guessed_name = guessed_name

                # set panel material
                if re.search('.*FP[0-9]{4}$', guessed_name):
                    product_series = guessed_name[-4]

                    if product_series == '8':  # check if strips
                        (self.window.Element('is_strips')
                         .Update(value=True))
                        self.set_to_strips()
                        self.reset_grid(is_horizontal=False)
                    else:
                        # if is already strips then set to vertical
                        if values['is_strips']:
                            (self.window.Element('is_vertical')
                             .Update(value=True))
                            self.set_to_panel()
                            self.reset_grid(is_horizontal=False)

                        if product_series == '2':  # check if
                            selective
                            (self.window.Element('is_selective')
                             .Update(value=True))
                        elif product_series == '3':  # check if TSS
                            (self.window.Element('is_tss')
                             .Update(value=True))
                        elif product_series == '4':  # check if
                            painted
                            (self.window.Element('is_painted')
                             .Update(value=True))

            # DEBUG
            # print('\nevent: ')
            # pprint(event)
            # print('\nvalues: ')
            # pprint(values)

        self.window.Close()
```

## drawing_utils.py

```
from math import pi, degrees, sqrt, sin, cos, atan2, floor, ceil
from numpy import add as np_add, array
from PIL import Image, ImageDraw, ImageFont
```



```
from pprint import pprint
import instructions as instr


class Drawing:
    def __init__(self):
        self.DRAWING_HEIGHT = 4500
        self.DRAWING_WIDTH = int(self.DRAWING_HEIGHT/sqrt(2))
        self.OO = (int(self.DRAWING_WIDTH/2) + 100,
                   int(self.DRAWING_HEIGHT/2))
        self.MM = int(self.DRAWING_HEIGHT/4500)  # 1 mm ~ 1 px
        self.THIN = int(3*self.MM)
        self.THICK = int(5*self.MM)
        self.FONT_SIZE = 130*self.MM
        self.FONT = ImageFont.truetype('arial.ttf', self.FONT_SIZE)
        self.FONT_S = ImageFont.truetype('arial.ttf',
                                         int(self.FONT_SIZE/2))
        self.FONT_XS = ImageFont.truetype('arial.ttf',
                                          int(5*self.FONT_SIZE/11))

        self.drawing = Image.new('RGB', size=(self.DRAWING_WIDTH,
                                              self.DRAWING_HEIGHT),
                                 color='white')
        self.draw = ImageDraw.Draw(self.drawing)

        self.test_values = {'absorber_info': 'test_absorber',
                            'down_left_exit_closed': False,
                            'down_right_exit_closed': True,
                            'head_to_head': 950.0,
                            'header_diameter': 22.0,
                            'header_exit_length': 60.0,
                            'header_length': 2000.0,
                            'is_copper': False,
                            'is_horizontal': True,
                            'is_meander': False,
                            'is_painted': False,
                            'is_selective': True,
                            'is_selective04': False,
                            'is_strips': False,
                            'is_vertical': False,
                            'meander_width': 900.0,
                            'panel_holes': True,
                            'panel_length': 2000.0,
                            'panel_width': 1000.0,
                            'riser_count': 20,
                            'riser_diameter': 8.0,
                            'riser_edge_distance': 70.0,
                            'riser_step': 100.0,
                            'strip_overlap': 10.0,
                            'up_left_exit_closed': False,
                            'up_right_exit_closed': False}

        self.test_settings = {'P1': 6900.0,
                              'P2': 7000,
                              'P3': 6700,
                              'P4': 8200,
                              'Z1': 5,
                              'Z2': 4,
                              'acc': 350,
                              'f1': 155,
                              'f2': 125,
                              'f3': 160,
                              'f4': 109,
                              'pres': 1.6,
```

```
                            'pulse1': 0.3,
                            'pulse2': 0.4,
                            'pulse3': 0.3,
                            'pulse4': 0.4,
                            'vel': 210.0}

    def add(self, point1, point2):
        return tuple(np_add(point1, point2))

    def xy(self, point):
        return self.add(self.OO, (point[0], point[1]))

    def clear_drawing(self):
        self.drawing = Image.new('RGB', size=(self.DRAWING_WIDTH,
                                              self.DRAWING_HEIGHT),
                                 color='white')
        self.draw = ImageDraw.Draw(self.drawing)

    def draw_dashed_line(self, start, end, dash, thickness=None,
                         color='black'):
        if thickness is None:
            thickness = self.THIN

        dl = sum(dash)
        ratios = [the_dash/dl for the_dash in dash]
        delta_x, delta_y = end[0] - start[0], end[1] - start[1]
        total_length = sqrt(delta_x**2 + delta_y**2)
        whole_parts = floor(total_length/dl)

        if whole_parts <= 0:
            return

        angle = atan2(delta_y, delta_x)
        dx, dy = dl*cos(angle), dl*sin(angle)

        # draw all whole parts
        part_end = start
        for i in range(whole_parts):
            for n in range(len(dash)):
                part_start = part_end
                part_end = self.add(part_start, (ratios[n]*dx,
                                                 ratios[n]*dy))

                if n % 2 == 0:
                    self.draw.line([part_start, part_end], color,
                                   thickness)

        # draw last part
        if total_length % dl != 0:
            part_start = self.add(start,
                                  (whole_parts*dx, whole_parts*dy))
            part_end = end

            self.draw.line([part_start, part_end], color, thickness
                )

    def draw_closed_exit(self, position, diameter, tube_angle,
                         thickness=None, color='black'):
        colors = (color, 'white')
        if thickness is None:
            thickness = self.THICK

        for i in range(2):
            t = (i - 0.5)*thickness
```



```
            self.draw.pieslice([self.add(position, (-diameter/2 + t
                ,
                                                    -diameter/2 + t
                                                      )),
                                self.add(position, (+diameter/2 - t
                                    ,
                                                    +diameter/2 - t
                                                      ))],
                               degrees(pi-tube_angle),
                               degrees(-tube_angle),
                               colors[i], None)

    def draw_tube(self, start, end, diameter, closed_exit=(False,
        False),
                  thickness=None, color='black'):
        if thickness is None:
            thickness = self.THICK

        if start == end:
            return

        dx, dy = end[0] - start[0], end[1] - start[1]
        phi = atan2(dx, dy)
        x0l = start[0] - diameter/2*cos(phi)
        x0r = start[0] + diameter/2*cos(phi)
        y0l = start[1] + diameter/2*sin(phi)
        y0r = start[1] - diameter/2*sin(phi)
        x1l = x0l + dx
        x1r = x0r + dx
        y1l = y0l + dy
        y1r = y0r + dy

        # draw body
        self.draw.rectangle([(x0l, y0l), (x1r, y1r)], 'white',
            thickness)
        self.draw.line([(x0l, y0l), (x1l, y1l)], color, thickness)
        self.draw.line([(x0r, y0r), (x1r, y1r)], color, thickness)

        # draw exits
        if closed_exit[0]:
            self.draw_closed_exit(start, diameter, phi, thickness,
                color)

        if closed_exit[1]:
            self.draw_closed_exit(end, diameter, pi+phi, thickness,
                color)

        self.draw.line([(x0l, y0l), (x0r, y0r)], color, thickness)
        self.draw.line([(x1l, y1l), (x1r, y1r)], color, thickness)

    def draw_curve(self, center, variant, radius, diameter,
                   thickness=None, color='black'):
        colors = (color, 'white')
        if thickness is None:
            thickness = self.THICK

        theta2 = 90*variant
        theta1 = theta2 - 90

        for j in range(2):
            for i in range(2):
                r = radius + (0.5 - j)*diameter + (0.5 - i)*
                    thickness
                start = self.add(center, (-r, -r))
```

```
                end = self.add(center, (+r, +r))

                self.draw.pieslice([start, end], theta1, theta2,
                                   colors[i], None)

    def draw_holes(self, panel_length, panel_width, lx, ly, radius):
        colors = ('black', 'white')

        # draw up-left and down-right holes
        for j in range(2):
            n = 2*j - 1
            for i in range(2):
                t = (0.5 - i)*self.THICK
                start = self.xy((n*(panel_width/2 - lx) - radius - t,
                                 n*(panel_length/2 - ly) - radius - t))
                end = self.xy((n*(panel_width/2 - lx) + radius + t,
                               n*(panel_length/2 - ly) + radius + t))

                self.draw.ellipse([start, end], colors[i], None)

    def draw_panel(self, panel_length, panel_width, has_holes=False):
        colors = ('black', 'white')

        for i in range(2):
            t = (0.5 - i)*self.THICK
            start = self.xy((-panel_width/2 - t, -panel_length/2 - t))
            end = self.xy((panel_width/2 + t, panel_length/2 + t))

            self.draw.rectangle([start, end], colors[i], None)

        if has_holes:
            lx = 50*self.MM
            ly = lx
            radius = 12*self.MM
            self.draw_holes(panel_length, panel_width, lx, ly, radius)

    def draw_strips(self, length, total_width, strip_step, strip_count):
        overlap = total_width - strip_count*strip_step
        self.draw_panel(length, total_width, has_holes=False)

        for i in range(1, strip_count):
            start1 = self.xy((-total_width/2 + i*strip_step, -length/2))
            end1 = self.add(start1, (0, length))
            start2 = self.add(start1, (overlap, 0))
            end2 = self.add(start2, (0, length))

            dash = (20*self.MM, 20*self.MM)
            self.draw.line([start1, end1], 'black', self.THICK)
            self.draw_dashed_line(start2, end2, dash)

    def draw_headers(self, diameter, head_to_head, header_length,
                     panel_width, exit_length, closed_exit,
                     is_horizontal):
        if isinstance(header_length, str):
            header_length = 0
```



```
        if is_horizontal:
            closed_exit = [closed_exit[i] for i in [0, 2, 1, 3]]

        for i in range(2):
            open_length = panel_width/2 + exit_length
            closed_length = header_length - open_length

            if closed_exit[2*i]:
                length_start = -closed_length
                length_end = open_length
            elif closed_exit[1+2*i]:
                length_start = -open_length
                length_end = closed_length
            else:
                length_start = -open_length
                length_end = open_length

            if is_horizontal:
                start = self.xy(((2*i-1)*head_to_head/2,
                                 length_start))
                end = self.xy(((2*i-1)*head_to_head/2, length_end))
            else:
                start = self.xy((length_start, (2*i-1)*head_to_head/2))
                end = self.xy((length_end, (2*i-1)*head_to_head/2))

            self.draw_tube(start, end, diameter, closed_exit[2*i:2*i+2])

    def draw_riser_grid(self, start_x, length, riser_count, step,
                        diameter, direction='v'):
        for i in range(riser_count):
            x = start_x - i*step
            y = length/2

            if direction.lower() == 'v':
                riser_start = self.xy((x, -y))
                riser_end = self.xy((x, y))
            elif direction.lower() == 'h':
                riser_start = self.xy((-y, x))
                riser_end = self.xy((y, x))

            self.draw_tube(riser_start, riser_end, diameter)

    def draw_riser_meander(self, edge_distance, meander_width,
                           length, riser_count, width, step,
                           diameter):
        mm = self.MM
        def add(point1, point2): return self.add(point1, point2)
        def xy(point): return self.xy(point)
        link_thickness = 12
        radius = 30
        if step < 2*radius:
            radius = step/2
        offset = 77

        for i in range(riser_count):
            riser_start = xy(((-width/2 + radius + edge_distance)*mm,
                              (length/2 - offset - i*step)*mm))
            riser_end = add(riser_start, ((meander_width - 2*radius)*mm,
                                          0))
```

```
            # draw first link
            if i == 0:
                link_start = add(riser_end, (-link_thickness/2*mm,
                                             offset*mm))
                link_end = add(link_start, (0, -offset*mm))
                self.draw_tube(link_start, link_end, link_thickness
                    *mm)

            # draw curved segments
            n = i % 2
            nradius = (1 - 2*n)*radius
            self.draw_curve(add(riser_start, (0, nradius*mm)), 3-n,
                            radius*mm, diameter*mm)
            self.draw_curve(add(riser_end, (0, -nradius*mm)), 1+3*n
                ,
                            radius*mm, diameter*mm)

            # draw horizontal riser segments
            self.draw_tube(riser_start, riser_end, diameter*mm)

            # draw verical riser segments
            ver_start = add(riser_end, ((radius - n*meander_width)*
                mm,
                                        -radius*mm))
            if i != riser_count - 1:
                ver_end = add(ver_start, (0, (-step + 2*radius)*mm)
                    )
            else:
                ver_end = xy((0, -length/2*mm))
                ver_end = (ver_start[0], ver_end[1])

            self.draw_tube(ver_start, ver_end, diameter*mm)

        # draw last link
        distance = (2*n-1)*(meander_width - radius - link_thickness
            /2)
        link2_start = add(ver_start, (distance*mm,
                                      (radius-diameter/2)*mm))
        link2_end = add(ver_end, (distance*mm, 0))

        self.draw_tube(link2_start, link2_end, link_thickness*mm)

        # draw inlet
        inlet_start = xy(((-width/2 + edge_distance)*mm, length/2*
            mm))
        inlet_end = add(inlet_start, (0, (-offset + radius)*mm))

        self.draw_tube(inlet_start, inlet_end, diameter*mm)

    def draw_risers(self, diameter, edge_distance, riser_length,
                    riser_count, total_width, step, is_horizontal,
                    is_meander, meander_width):
        if is_meander:
            self.draw_riser_meander(edge_distance, meander_width,
                                    riser_length, riser_count,
                                    total_width, step, diameter)
            return
        elif is_horizontal:
            direction = 'h'
        else:
            direction = 'v'

        start = total_width/2 - edge_distance
        self.draw_riser_grid(start, riser_length, riser_count, step
```



```
            ,
                             diameter, direction)

    def draw_arrow(self, start, mid, end, color='black', thickness=
        None,
                   arrow_count=2):
        if thickness is None:
            thickness = self.THICK
        arrowhead_width = 7*thickness
        arrowhead_length = 1.5*arrowhead_width
        dx, dy = mid[0] - start[0], mid[1] - start[1]
        theta = atan2(dy, dx)
        rot = array([[cos(theta), -sin(theta)],
                     [sin(theta), cos(theta)]])
        if sqrt(dx**2 + dy**2) < 2.5*arrowhead_length:
            is_in = -1
        else:
            is_in = 1

        # set each arrowhead's 3-points
        point = [[], []]
        point[0] = [start, [], []]
        point[1] = [mid, [], []]
        for the_arrow in range(2):
            n = 1 - 2*the_arrow
            for the_point in range(1, 3):
                m = 3 - 2*the_point
                du = array([n*is_in*arrowhead_length,
                            m*arrowhead_width/2])
                rot_du = tuple(rot.dot(du))
                point[the_arrow][the_point] = \
                    self.add(point[the_arrow][0], rot_du)

            # print('arrow ' + str(the_arrow) + ':')
            # pprint(point[the_arrow])

            # draw arrowhead
            if the_arrow < arrow_count:
                self.draw.polygon(point[the_arrow], color, None)

        end2 = end
        start1 = ((point[0][1][0] + point[0][2][0])/2,
                  (point[0][1][1] + point[0][2][1])/2)
        start2 = ((point[1][1][0] + point[1][2][0])/2,
                  (point[1][1][1] + point[1][2][1])/2)
        if arrow_count == 1:
            start2 = mid
        if is_in == -1:
            du = array([-arrowhead_length, 0])
            rot_du = tuple(rot.dot(du))
            end1 = self.add(start1, rot_du)
        elif is_in == 1:
            end1 = start2

        # draw arrow line and dimension line
        self.draw.line([start1, end1], color, thickness)
        self.draw.line([start2, end2], color, thickness)

    def draw_dimension(self, point1, point2, label, steps='',
                       extra_offset=0, color='blue', arrows=2):
        def add(point1, point2): return self.add(point1, point2)
        FONT = self.FONT
        FONT_S = self.FONT_S
        FONT_SIZE = self.FONT_SIZE
```

```
        ARR_LENGTH = 10*self.THICK

        # if the length is 0 don't draw dimension
        if point1 == point2:
            return

        offset = 2*ARR_LENGTH + extra_offset

        # determine textbox dimensions
        label_size = self.draw.textsize(label, FONT)
        steps_size = self.draw.textsize(steps, FONT_S)
        text_size = (label_size[0] + steps_size[0],
                     max(label_size[1], steps_size[1]))

        # define dimension line parameters
        start, mid = point1, point2
        dx, dy = mid[0] - start[0], mid[1] - start[1]
        theta = atan2(dy, dx)
        dl = (cos(theta)*text_size[0], sin(theta)*text_size[1])
        offset_dl = add(dl, (cos(theta)*offset, sin(theta)*offset))

        # determine if label fits inside dimension and set end
           point
        if sqrt(dl[0]**2+dl[1]**2) + 2*ARR_LENGTH < sqrt(dx**2+dy
           **2):
            # is in
            end = ((start[0] + mid[0])/2, ((start[1] + mid[1])/2))
            text_xy = add(end, (-abs(dl[0]/2), abs(dl[1]/2) -
              FONT_SIZE))
        else:
            # is out
            end = add(mid, offset_dl)
            if theta == 0:
                text_xy = add(end, (-text_size[0], -FONT_SIZE))
            elif 0 < theta < pi/2:
                text_xy = add(end, (0, -FONT_SIZE))
            elif theta == pi/2:
                text_xy = add(end, (0, -FONT_SIZE))
            elif pi/2 < theta < pi:
                text_xy = add(end, (-text_size[0], -FONT_SIZE))
            elif theta == pi or theta == -pi:
                text_xy = add(end, (0, -FONT_SIZE))
            elif -pi < theta < -pi/2:
                text_xy = add(end, (-text_size[0], -FONT_SIZE))
            elif theta == -pi/2:
                text_xy = add(end, (0, 0))
            elif -pi/2 < theta < 0:
                text_xy = add(end, (0, -FONT_SIZE))

        self.draw_arrow(start, mid, end, color, arrow_count=arrows)

        self.draw.text(text_xy, label, color, FONT)
        self.draw.text(add(text_xy, (label_size[0] + 15*self.MM, 0)
           ),
                       steps, 'black', FONT_S)

    def write_instruction(self, operator, location, n, m, color='
       black',
                          font=None):
        if font is None:
            if operator == 1:
                font = self.FONT_S
            elif operator == 2:
                font = self.FONT_S
```



```

        text = ' ' + instr.step[n][m]
        if m != 0:
            text = str(m) + '.' + text
        if n != 0:
            text = str(n) + '.' + text

        if operator == 1:
            self.draw.text(location, text, color, font)
        elif operator == 2:
            self.draw.text(location, text, color, font)

    def make_labels_and_steps(self, values):
        labels = {}
        for key, value in values.items():
            if key == 'absorber_info':
                labels[key] = value
            else:
                labels[key] = str(value).rstrip('0').rstrip('.')

        steps = {
            'riser_step': '1.6, 6.2,\n2.4.2.3',
            'header_exit_length': '4.3.2.2\n7.1',
            'riser_edge_distance': '4.3.2.2\n7.2',
            'table_edge_distance': '2.1.2',
            'strip_step': '',
            'panel_residual': '4.3.1.1.4',
            'diagonals': '7.3'
        }

        return labels, steps

    def make_base_drawing(self, values):
        mm = self.MM

        # define parameters
        closed_exit = (values['up_left_exit_closed'],
                       values['up_right_exit_closed'],
                       values['down_left_exit_closed'],
                       values['down_right_exit_closed'])

        if values['is_horizontal']:
            total_width = values['panel_length']
        else:
            total_width = values['panel_width']

        # draw panel/strips
        if values['is_strips']:
            self.draw_strips(values['panel_length']*mm,
                             values['panel_width']*mm,
                             values['riser_step']*mm,
                             values['riser_count'])
        else:
            self.draw_panel(values['panel_length']*mm,
                            values['panel_width']*mm,
                            has_holes=values['panel_holes'])

        # draw risers
        self.draw_risers(values['riser_diameter']*mm,
                         values['riser_edge_distance']*mm,
                         (values['head_to_head'] -
                          values['header_diameter'])*mm,
                         values['riser_count'],
                         total_width*mm,
```

```
                        values['riser_step']*mm,
                        values['is_horizontal'],
                        values['is_meander'],
                        values['meander_width'])

        # draw headers
        self.draw_headers(values['header_diameter']*mm,
                          values['head_to_head']*mm,
                          values['header_length']*mm,
                          total_width*mm,
                          values['header_exit_length']*mm,
                          closed_exit, values['is_horizontal'])

    def make_dimensions(self, values):
        mm = self.MM
        def xy(point): return self.xy(point)
        def add(point1, point2): return self.add(point1, point2)
        CHANNEL_LEN = 250
        CHANNEL_WID = 7
        TABLE_WIDTH = 1533
        DISTANCE = 175

        # set necessary parameters
        if values['is_horizontal']:
            total_width = values['panel_length']
            total_length = values['panel_width']
        else:
            total_width = values['panel_width']
            total_length = values['panel_length']

        closed_exit = [values['up_right_exit_closed'],
                       values['down_right_exit_closed'],
                       values['down_left_exit_closed'],
                       values['up_left_exit_closed']]

        if values['is_horizontal']:
            closed_exit = [closed_exit[i] for i in [2, 1, 0, 3]]

        labels, steps = self.make_labels_and_steps(values)

        # draw upper/lower panel residual's length
        panel_residual = (total_length - values['head_to_head'] -
                          values['header_diameter'])/2
        residual_label = str(abs(panel_residual)).rstrip('0').
           rstrip('.')

        if closed_exit[1:3] == [False, False]:
            n = 1
        else:
            n = -1
        point2 = (0, n*total_length/2*mm)
        point1 = add(point2, (0, -n*panel_residual*mm))

        if panel_residual < 0:
            point1, point2 = point2, point1

        if values['is_horizontal']:
            point1 = (point1[1], point1[0])
            point2 = (point2[1], point2[0])

        self.draw_dimension(xy(point1), xy(point2), residual_label,
                            steps['panel_residual'],
                            extra_offset=-75*mm)
```



```
        # draw diagonal's quality check
        if not (closed_exit[0:2] == [True, True] or
                closed_exit[2:4] == [True, True]):
            point1 = (-(total_width/2 + values['header_exit_length'
               ])*mm,
                      (values['head_to_head'] -
                       values['header_diameter'])/2*mm)
            point2 = (-point1[0], -point1[1])

            if values['is_horizontal']:
                point1 = (point1[1], point1[0])
                point2 = (point2[1], point2[0])

            if closed_exit[0] or closed_exit[2]:
                point1 = (-point1[0], point1[1])
                point2 = (-point2[0], point2[1])
        else:
            if values['is_vertical'] or values['is_strips']:
                point2 = ((total_width/2 + values['riser_diameter'
                   ]/2 -
                           values['riser_edge_distance'])*mm,
                           -(values['head_to_head'] -
                             values['header_diameter'])/2*mm)
                point1 = (point2[0] -
                           ((values['riser_count']-1)*values['
                              riser_step']
                            + values['riser_diameter'])*mm,
                           -point2[1])

        label = (str(round(sqrt((2*point1[0])**2+(2*point1[1])**2),
                           1)).rstrip('0').rstrip('.'))
        self.draw_dimension(xy(point1), xy(point2), label,
                            steps['diagonals'])

        # draw header exit length
        if closed_exit[1]:
            n = -1
        else:
            n = 1

        l_start = (n*(total_width/2 + values['header_exit_length'])
           *mm,
                   (values['head_to_head']/2)*mm -
                   values['header_diameter']/2 - self.THICK)
        l_end = add(l_start, (0, -70*mm))
        point2 = add(l_start, (0, -50*mm))
        point1 = add(point2, (-n*values['header_exit_length']*mm,
           0))

        if values['is_horizontal']:
            l_start = (l_start[1], l_start[0])
            l_end = (l_end[1], l_end[0])
            point1 = (point1[1], point1[0])
            point2 = (point2[1], point2[0])

        self.draw.line([xy(l_start), xy(l_end)], 'blue', self.THIN)
        self.draw_dimension(xy(point1), xy(point2),
                            labels['header_exit_length'],
                            steps['header_exit_length'])

        # draw riser-edge distance
        point2 = (total_width/2*mm, -total_length/3*mm)
        point1 = add(point2, (-values['riser_edge_distance']*mm, 0)
           )
```

```
663
664         if values['is_meander']:
665             point1 = (-point1[0], -point1[1])
666             point2 = (-point2[0], -point2[1])
667             l_start = add(point1, (0, -values['riser_step']/2*mm))
668             l_end = add(point1, (0, values['riser_step']/2*mm))
669             self.draw.line([xy(l_start), xy(l_end)], 'blue',
670                            self.THIN)
671         elif values['is_horizontal']:
672             point1 = (point1[1], point1[0])
673             point2 = (point2[1], point2[0])
674
675         self.draw_dimension(xy(point1), xy(point2),
676                             labels['riser_edge_distance'],
677                             steps['riser_edge_distance'],
678                             extra_offset=60*mm)
679
680         # draw riser step
681         if values['is_meander']:
682             point2 = ((-values['head_to_head'] +
683                        (values['head_to_head'] - 77) %
684                        values['riser_step'])*mm, 0)
685             point1 = add(point2, (values['riser_step']*mm, 0))
686         else:
687             point2 = ((total_width/2 - values['riser_edge_distance'
                    ])*mm,
688                       -total_length/6*mm)
689             point1 = add(point2, (-values['riser_step']*mm, 0))
690
691             if values['is_horizontal']:
692                 point1 = (point1[1], point1[0])
693                 point2 = (point2[1], point2[0])
694
695         self.draw_dimension(xy(point1), xy(point2), labels['
                riser_step'],
696                             steps['riser_step'])
697
698         # draw distance from table's right edge
699         if values['is_vertical'] or values['is_strips']:
700             for j in range(2):  # Y-axis symmetry
701                 if TABLE_WIDTH - values['panel_width'] < DISTANCE:
702                     distance = (TABLE_WIDTH-values['panel_width'])
                            /2
703                 else:
704                     distance = DISTANCE
705
706                 label = str(abs(distance)).rstrip('0').rstrip('.')
707                 step = steps['table_edge_distance']
708
709                 channel_mid = ((1-2*j)*(total_width/2 + distance)*
                        mm,
710                                total_length/4*mm)
711                 channel_start = add(channel_mid,
712                                     (0, -CHANNEL_LEN/2*mm))
713                 channel_end = add(channel_mid, (0, CHANNEL_LEN/2*mm
                        ))
714                 point1 = (channel_mid[0] - (1-2*j)*distance*mm,
715                           channel_mid[1])
716                 point2 = add(point1, ((1-2*j)*distance*mm, 0))
717
718                 if distance < 0:
719                     point1, point2 = point2, point1
720                 elif distance >= DISTANCE and j == 1:
721                     point2 = point1
```



```
722                 channel_end = channel_start
723
724             self.draw_tube(xy(channel_start), xy(channel_end),
725                            CHANNEL_WID*mm, thickness=self.THIN,
726                            color='blue')
727             self.draw_dimension(xy(point1), xy(point2), label,
728                                 step)
729
730         # draw strip width
731         if values['is_strips']:
732             strip_width = (total_width - (values['riser_count']-1)
                    *
733                            values['riser_step'])
734             point2 = (-total_width/2*mm, -total_length/4*mm)
735             point1 = add(point2, (strip_width*mm, 0))
736             label = (str(strip_width).rstrip('0').rstrip('.'))
737             self.draw_dimension(xy(point1), xy(point2), label,
738                                 steps['strip_step'], color='black')
739
740         # set offsets
741         if values['is_horizontal']:
742             offset_h = 60
743             offset_v = 750
744         else:
745             offset_h = 300
746             offset_v = 360
747
748         # draw closed header length:
749         if closed_exit[1:3] == [False, False]:
750             n = 1
751         else:
752             n = -1
753
754         if isinstance(values['header_length'], str):
755             values['header_length'] = 0
756         open_length = total_width/2 + values['header_exit_length']
757         closed_length = values['header_length'] - open_length
758
759         if closed_exit[int((n+5)/2)]:
760             length_left = -closed_length - values['header_diameter'
                    ]/2
761             length_right = open_length
762             label = str(values['header_length']).rstrip('0').rstrip
                    ('.')
763         elif closed_exit[int((1-n)/2)]:
764             length_left = -open_length
765             length_right = closed_length + values['header_diameter'
                    ]/2
766             label = str(values['header_length']).rstrip('0').rstrip
                    ('.')
767         else:
768             length_left = -open_length
769             length_right = open_length
770             label = (str(round(total_width +
771                      2*values['header_exit_length'], 1))
772                      .rstrip('0').rstrip('.'))
773
774         l_1_start = (length_left*mm, n*((-values['head_to_head'] -
775                      values['header_diameter'])/2*mm - self.THICK))
776         l_1_end = add(l_1_start, (0, -n*offset_v*mm))
777         l_2_start = (length_right, l_1_start[1])
778         l_2_end = (length_right, l_1_end[1])
779         point1 = add(l_1_end, (0, n*50*mm))
780         point2 = add(l_2_end, (0, n*50*mm))
```

```
781
782         if values['is_horizontal']:
783             l_1_start = (l_1_start[1], l_1_start[0])
784             l_1_end = (l_1_end[1], l_1_end[0])
785             l_2_start = (l_2_start[1], l_2_start[0])
786             l_2_end = (l_2_end[1], l_2_end[0])
787             point1 = (point1[1], point1[0])
788             point2 = (point2[1], point2[0])
789
790         self.draw.line([xy(l_1_start), xy(l_1_end)], 'black', self.
               THIN)
791         self.draw.line([xy(l_2_start), xy(l_2_end)], 'black', self.
               THIN)
792         self.draw_dimension(xy(point1), xy(point2), label, color='
               black')
793
794         # draw head-to-head distance
795         dash_1_start = ((-total_width/2 + 150)*mm,
796                         -values['head_to_head']/2*mm)
797         dash_1_end = ((-total_width/2 - values['header_exit_length'
               ] -
798                        50)*mm, dash_1_start[1])
799         l_1_start = dash_1_end
800         l_1_end = add(l_1_start, (-offset_h*mm, 0))
801         dash_2_start = (dash_1_start[0], -dash_1_start[1])
802         dash_2_end = (dash_1_end[0], -dash_1_end[1])
803         l_2_start = (l_1_start[0], -l_1_start[1])
804         l_2_end = (l_1_end[0], -l_1_end[1])
805         point1 = add(l_1_end, (50*mm, 0))
806         point2 = add(l_2_end, (50*mm, 0))
807
808         if values['is_horizontal']:
809             dash_1_start = (dash_1_start[1], dash_1_start[0])
810             dash_1_end = (dash_1_end[1], dash_1_end[0])
811             dash_2_start = (dash_2_start[1], dash_2_start[0])
812             dash_2_end = (dash_2_end[1], dash_2_end[0])
813             l_1_start = (l_1_start[1], l_1_start[0])
814             l_1_end = (l_1_end[1], l_1_end[0])
815             l_2_start = (l_2_start[1], l_2_start[0])
816             l_2_end = (l_2_end[1], l_2_end[0])
817             point1 = (point1[1], point1[0])
818             point2 = (point2[1], point2[0])
819
820         label = str(values['head_to_head']).rstrip('0').rstrip('.')
821         dash = (80*mm, 10*mm, 10*mm, 10*mm)
822         self.draw_dashed_line(xy(dash_1_start), xy(dash_1_end),
               dash,
823                               self.THIN)
824         self.draw_dashed_line(xy(dash_2_start), xy(dash_2_end),
               dash,
825                               self.THIN)
826         self.draw.line([xy(l_1_start), xy(l_1_end)], 'black', self.
               THIN)
827         self.draw.line([xy(l_2_start), xy(l_2_end)], 'black', self.
               THIN)
828         self.draw_dimension(xy(point1), xy(point2), label, color='
               black')
829
830         # set offsets
831         if values['is_horizontal']:
832             offset_h = 360
833             offset_v = 400
834         else:
835             offset_h = 750
```



```
836             offset_v = 200
837
838         # draw panel width
839         if closed_exit[1:3] == [False, False]:
840             n = 1
841         else:
842             n = -1
843
844         label = str(total_width).rstrip('0').rstrip('.')
845         l_1_start = (-total_width/2*mm,
846                      -n*(total_length/2*mm-2*self.THIN))
847         l_1_end = add(l_1_start, (0, -n*offset_v*mm))
848         l_2_start = (-l_1_start[0], l_1_start[1])
849         l_2_end = (-l_1_end[0], l_1_end[1])
850         point1 = add(l_1_end, (0, n*50*mm))
851         point2 = add(l_2_end, (0, n*50*mm))
852
853         if values['is_horizontal']:
854             l_1_start = (l_1_start[1], l_1_start[0])
855             l_1_end = (l_1_end[1], l_1_end[0])
856             l_2_start = (l_2_start[1], l_2_start[0])
857             l_2_end = (l_2_end[1], l_2_end[0])
858             point1 = (point1[1], point1[0])
859             point2 = (point2[1], point2[0])
860
861         self.draw.line([xy(l_1_start), xy(l_1_end)], 'black', self.
               THIN)
862         self.draw.line([xy(l_2_start), xy(l_2_end)], 'black', self.
               THIN)
863         self.draw_dimension(xy(point1), xy(point2), label, color='
               black')
864
865         # draw panel length
866         label = str(total_length).rstrip('0').rstrip('.')
867         l_1_start = (-total_width/2*mm-2*self.THIN, -total_length
               /2*mm)
868         l_1_end = add(l_1_start, (-offset_h*mm, 0))
869         l_2_start = (l_1_start[0], -l_1_start[1])
870         l_2_end = (l_1_end[0], -l_1_end[1])
871         point1 = add(l_1_end, (50*mm, 0))
872         point2 = add(l_2_end, (50*mm, 0))
873
874         if values['is_horizontal']:
875             l_1_start = (l_1_start[1], l_1_start[0])
876             l_1_end = (l_1_end[1], l_1_end[0])
877             l_2_start = (l_2_start[1], l_2_start[0])
878             l_2_end = (l_2_end[1], l_2_end[0])
879             point1 = (point1[1], point1[0])
880             point2 = (point2[1], point2[0])
881
882         self.draw.line([xy(l_1_start), xy(l_1_end)], 'black', self.
               THIN)
883         self.draw.line([xy(l_2_start), xy(l_2_end)], 'black', self.
               THIN)
884         self.draw_dimension(xy(point1), xy(point2), label, color='
               black')
885
886     def make_order_info(self, values, settings):
887         FONT, FONT_S = self.FONT, self.FONT_S
888         FONT_SIZE = self.FONT_SIZE
889         SPCING = int(self.FONT_SIZE/5)
890
891         # draw absorber info
892         label = values['absorber_info']
```

```
        text_size = self.draw.textsize(label, FONT)
        text_xy = ((self.DRAWING_WIDTH - text_size[0])/2, FONT_SIZE
            )
        self.draw.text(text_xy, label, 'black', FONT, spacing=
            SPCING)

        # draw coordinate system
        x_label = ' I'
        y_label = 'II'
        if values['is_horizontal']:
            x_label, y_label = y_label, x_label

        origin_xy = (3*self.FONT_SIZE,
                     self.DRAWING_HEIGHT - 4*self.FONT_SIZE)
        self.draw_dimension(self.add(origin_xy, (400, 0)),
            origin_xy,
                            x_label, color='black', arrows=1)
        self.draw_dimension(self.add(origin_xy, (0, -400)),
            origin_xy,
                            y_label, color='black', arrows=1)

        # print material info
        label = ''
        # if values['is_selective']:
        #     panel_material_text = επιλεκτικό
        # elif values['is_selective04']:
        #     panel_material_text = επιλεκτικό 0.4
        # elif values['is_painted']:
        #     panel_material_text = βαμμένο
        # elif values['is_copper']:
        #     panel_material_text = από χαλκό
        # if values['is_strips']:
        #     panel_type_text = str(values['riser_count']) +
            φινάκια
        # else:
        #     panel_type_text = Φύλλο
        # label = (panel_type_text + ': ' + panel_material_text + '
            , ' +
        #          str(round(values['panel_length'])) + 'X' +
        #          str(round(values['panel_width'])))
        label += ('\n' + Σωλήνες:  Φ +
                  str(round(values['header_diameter'])) + , Φ +
                  str(round(values['riser_diameter'])))
        text_size = self.draw.textsize(label, FONT_S)
        text_xy = (FONT_SIZE,
                   self.DRAWING_HEIGHT - text_size[1] - FONT_SIZE)
        self.draw.text(text_xy, label, 'black', FONT_S, spacing=
            SPCING)

        # print laser & machine settings info
        setting = {}
        if values['riser_diameter'] <= 8:
            setting['Z'] = settings['Z1']
        else:
            setting['Z'] = settings['Z2']
        if values['is_selective']:
            for item in ('P', 'pulse', 'f'):
                setting[item] = settings[item + '1']
        if values['is_selective04']:
            for item in ('P', 'pulse', 'f'):
                setting[item] = settings[item + '2']
        if values['is_tss']:
            for item in ('P', 'pulse', 'f'):
                setting[item] = settings[item + '3']
```



```
        if values['is_painted']:
            for item in ('P', 'pulse', 'f'):
                setting[item] = settings[item + '4']
        if values['is_copper']:
            for item in ('P', 'pulse', 'f'):
                setting[item] = settings[item + '5']

        label_l = ('VEL: ' + str(settings['vel']) + '\n' +
                   'ACC: ' + str(settings['acc']) + '\n' +
                   'PRES: ' + str(settings['pres']) + '\n' +
                   'Z: ' + str(setting['Z']) + '\n')
        label_r = ('POWER: ' + str(setting['P']) + '\n' +
                   'PULSE: ' + str(setting['pulse']) + '\n' +
                   'FREQ: ' + str(setting['f']))
        text_l_size = self.add(self.draw.textsize(label_l, FONT_S),
                               (int(FONT_SIZE/2), 0))
        text_r_size = self.draw.textsize(label_r, FONT_S)
        text_size = (text_l_size[0] + text_r_size[0],
                     max(text_l_size[1], text_r_size[1]))
        text_xy = (self.DRAWING_WIDTH - FONT_SIZE - text_size[0],
                   self.DRAWING_HEIGHT - FONT_SIZE - text_size[1])
        self.draw.text(text_xy, label_l, 'black', FONT_S, spacing=
            SPCING)
        self.draw.text(self.add(text_xy, (text_l_size[0], 0)),
            label_r,
                       'black', FONT_S, spacing=SPCING)

    def make_order_instructions_page_1(self, values):
        # write instructions for operator 1
        location = [2*self.FONT_SIZE, 2*self.FONT_SIZE]

        self.draw.text(location, Χρήσιμα βήματα:, 'black', self.FONT)
        location[1] += int(2.5*self.FONT_SIZE)

        for n in range(1, 8):
            if n == 5 and values['riser_diameter'] == 8:
                continue
            self.write_instruction(1, location, n, 0)
            location[1] += 2*self.FONT_SIZE

    def make_order_instructions_page_2(self, values):
        # write instructions for operator 2
        location = [self.FONT_SIZE, self.FONT_SIZE]

        self.draw.text(location, Χρήσιμα υπο-βήματα:, 'black',
                       self.FONT)
        location[1] += int(2.5*self.FONT_SIZE)

        for n in range(1, 8):
            if n == 5 and values['riser_diameter'] == 8:
                continue

            for m in range(len(instr.step[n])):
                # n = 1
                if n == 1:
                    if not values['panel_holes']:
                        if m == 3:
                            continue
                    if not values['is_meander']:
                        if m == 5:
                            continue
                    if not values['is_strips']:
                        if m in (6, 7):
                            continue
```

```
                # n = 2
                if n == 2 and not values['is_strips']:
                    if m in (3, 4, 5):
                        continue
                elif n == 2 and values['is_strips']:
                    if m in (1, 2):
                        continue

                # n = 3
                if n == 3 and (values['is_vertical'] or
                               values['is_strips']):
                    if m in (3, 4, 5):
                        continue
                elif n == 3 and not (values['is_vertical'] or
                                     values['is_strips']):
                    if m == 2:
                        continue

                # skip super-steps
                if m != 0:
                    self.write_instruction(2, location, n, m)
                if not (n == 1 and m == 0):
                    location[1] += self.FONT_SIZE/1.4

                if m == 0:
                    if n in (1, 7):
                        hard_order_text = (Χωρίς αυστηρή διαδοχή)
                    else:
                        hard_order_text = (Με αυστηρή διαδοχή)

                    location[1] -= 0.5*self.FONT_SIZE/1.3
                    self.draw.text(location, hard_order_text, '
                       black',
                                   self.FONT_XS)
                    location[1] += self.FONT_SIZE/1.3

    def make_job_order(self, file_name, values, settings):
        self.make_base_drawing(values)
        self.make_dimensions(values)
        self.make_order_info(values, settings)
        self.drawing.save(file_name, 'PDF')

        self.clear_drawing()
        self.make_order_instructions_page_1(values)
        self.drawing.save(file_name, 'PDF', append=True)

        self.clear_drawing()
        self.make_order_instructions_page_2(values)
        self.drawing.save(file_name, 'PDF', append=True)


# TEST
# drawing = Drawing()
#
# drawing.make_job_order('C:\\Users\\George\\Desktop\\test.pdf',
#                        drawing.test_values,
#                        drawing.test_settings)
```

file_utils.py

```
from os import path, listdir, mkdir, rename
# from pprint import pprint

```



```

class FileManager:
    def __init__(self):
        self.drawing_number = '0000'
        self.dir_path = 'drawing_no-product_name.pdf'
        self.OLDS_DIR_NAME = 'old'

    def current_drawings(self, drawing_number, dir_path):
        cur_drawings = []
        if drawing_number == '':
            return cur_drawings
        for file_name in listdir(dir_path):
            if file_name.startswith(drawing_number):
                cur_drawings.append(file_name)
        return cur_drawings

    def move_file(self, file_name, old_dir, new_dir):
        # determine appropriate file name
        new_file_name = file_name
        count = 1
        name, extension = path.splitext(file_name)
        while path.isfile(path.join(new_dir, new_file_name)):
            count += 1
            new_file_name = name + ' (' + str(count) + ')' +
               extension

        # move the file
        old_file_path = path.join(old_dir, file_name)
        new_file_path = path.join(new_dir, new_file_name)
        rename(old_file_path, new_file_path)

    def prepare_dir(self, drawing_number, dir_path, olds_dir_name):
        # store any existing file names with same drawing number
        cur_drawings = self.current_drawings(drawing_number,
           dir_path)
        if cur_drawings:
            # if olds directory doesn't exist, create it
            olds_dir_path = dir_path + '\\' + olds_dir_name
            if not path.isdir(olds_dir_path):
                mkdir(olds_dir_path)

            old_drawings_path = olds_dir_path + '\\' +
               drawing_number
            if not path.isdir(old_drawings_path):
                mkdir(old_drawings_path)

            # move every existing drawing to olds dir without
               overwriting
            for drawing in cur_drawings:
                self.move_file(drawing, dir_path, old_drawings_path
                   )

            files_were_found = True
        else:
            files_were_found = False

        return files_were_found
```

instructions.py

```
step = [
    [Σετάρισμα.],
    [Προετοιμασία/συγκέντρωση εξοπλισμού.,
     Ελεγχος αν υπάρχουν λαστιχένιες ροδέλες στα παξιμάδια των οδηγών
```

```
5       της τράπεζας.,
6       Συγκέντρωση εργαλείων (κατσαβίδι/α, κλειδί, χαρτοταινία, και
7       χαρακάκι).,
8       Συγκέντρωση εργαλείων διάνοιξης οπών (σφυρί, ζουμπάς, μαρμάρινο
9       πλακίδιο, έλασμα με οπή).,
10      Απομάκρυνση των εργαλείων από την τράπεζα πριν την αλλαγή της.,
11      Συγκέντρωση των ξύλινων πλαισίων και των βαρών τους για τη στήριξη
12      του μαιάνδρου  + 'riser.',
13      Επιλογή μέτρου κατάλληλου μήκους = βήμα -  + 'overlap.',
14      Επιλογή φύλλου-λωρίδας κατάλληλου μήκους (ίσο με το συνολικό
15      πλάτος των φινακιών).],
16     [Τοποθέτηση στηρίξεων φύλλου.,
17      Τοποθέτηση του φύλλου αρχικά κατά την μεγάλη πλευρά του.,
18      Τοποθέτηση του φύλλου κατά τη μικρή πλευρά του.,
19      Τοποθέτηση οδηγών της κάτω πλευράς της τράπεζας.,
20      Τοποθέτηση των στοπ των φινακιών.,
21      Τοποθέτηση οδηγών της πάνω μπάρας (όμοια με 2.3).],
22     [Μεταφορά/Αλλαγή της πάνω μπάρας.,
23      Ξεβίδωμα των βιδών σε κάθε άκρο της πάνω μπάρας.,
24      Μεταφορά της πάνω μπάρας κατά τη διεύθυνση II.,
25      Αφαίρεση υπάρχουσας μπάρας (σε περίπτωση που δεν είναι η
26      κατάλληλη).,
27      Τοποθέτηση κατάλληλης μπάρας (για μαίανδρο:  πάνω μπάρα, για
28      οριζόντιο:  αριστερή μπάρα).,
29      Η μπάρα δεν βιδώνεται ακόμα.],
30     [Τοποθέτηση στηρίξεων του  + 'grid' +  σωλήνων.,
31      Αφαίρεση υπαρχόντων στοπ των  + 'header.',
32      Τοποθέτηση του grid (πρόχειρα) πάνω από το/τα φύλλο/α.,
33      Τοποθέτηση του πρώτου header (σε οριζόντιο:  ο αριστερός, αλλιώς:  ο
34      κάτω).,
35      Τοποθέτηση του δεύτερου  + 'header.'],
36     [Αλλαγή της ρόδας του  + 'laser.',
37      Ξεβίδωμα του παξιμαδιού της ρόδας.,
38      Αφαίρεση της βίδας, του παξιμαδιού, και της ρόδας.,
39      Τοποθέτηση της άλλης ρόδας, με την ίδια βίδα και το παξιμάδι.,
40      Βίδωμα του παξιμαδιού.,
41      Από Φ8 σε Φ15:  5 αντιωρολογιακές περιστροφές του  + 'knob.' +
42      Από Φ15 σε Φ8 το αντίστροφο.,
43      Δοκιμή και έλεγχος ομαλής περιστροφής της ρόδας.],
44     [Προετοιμασία του προγράμματος του  + 'laser.',
45      Επιλογή κατάλληλου προγράμματος για το συγκεκριμένο/παρόμοιο
46      προϊόν.,
47      Εισαγωγή νέου βήματος (ή λοιπών χαρακτηριστικών αν χρειάζεται).,
48      Καλιμπράρισμα του  + 'laser' +  με κατάλληλο  + 'offset.'],
49     [Ποιοτικός έλεγχος.,
50      Μέτρηση μήκους εξόδου  + 'header.',
51      Μέτρηση απόστασης πρώτου  +  'riser' +  από την άκρη του φύλλου.,
52      Μέτρηση των διαγωνίων του  + 'grid' +
53       ώστε να ελεγχθεί η καθετότητα των σωλήνων.,
54      Έλεγχος, οπτικός και με κατάλληλος εργαλείο, της συγκόλλησης.],
55     [Τακτοποίηση του εξοπλισμού.,
56      Τοποθέτηση κάθε εργαλείου που χρησιμοποιήθηκε πίσω στη θέση του.]
57 ]
```



# Παράρτημα Β Κώδικας προγράμματος ρύθμισης ι-διοσυσκευής

## main.py

```
1  from gui_utils import Gui
2
3
4  def main():
5      gui = Gui()
6      gui.run()
7
8
9  if __name__ == '__main__':
10     main()
```

## gui_utils.py

```
1  import sys
2  from os import environ, path
3  from configuration_utils import Configuration
4  from file_utils import FileManager
5
6  # from pprint import pprint
7
8  if sys.version_info[0] >= 3:
9      import PySimpleGUI as sg
10 else:
11     import PySimpleGUI27 as sg
12
13
14 class Gui:
15     def __init__(self):
16         LCOL_WIDTH = 16
17         FRAME_LCOL_WIDTH = 13
18         OFFSETS_LCOL_WIDTH = 11
19         TB_WID = 8
20         self.was_horizontal = False
21
22         HOMEPATH = 'C:' + environ['HOMEPATH']
23         if getattr(sys, 'frozen', False):
24             CURRENT_DIR = path.dirname(path.realpath(sys.executable
                   ))
25         elif __file__:
26             CURRENT_DIR = path.dirname(path.realpath(__file__))
27         self.guessed_name = 'configuration'
28         self.default_output_folder = HOMEPATH + '\\'
29         self.output_file = (self.default_output_folder +
30                             self.guessed_name + '.dat')
31         self.input_file = self.output_file
32         self.OLDS_DIR = 'old'
33         self.OFFSETS_FILE = CURRENT_DIR + '\\offsets.cfg'
34         self.OFFSETS_FILE_EXISTS = path.isfile(self.OFFSETS_FILE)
35         self.CONFIG_FILE = CURRENT_DIR + '\\configuration.txt'
36
37         # machine default offsets
38         self.default_offsets = {
39             'motor_1': 0,
40             'motor_2': 0,
41             'motor_3': 0,
42             'motor_4': 0,
43             'motor_5': 0,
44             'motor_6': 0,
```

```
            'motor_7': 0,
            'motor_8': 0
        }

        # offsets in use
        self.offsets = self.default_offsets.copy()

        # visual options
        sg.SetOptions(
            icon=None,
            button_color=(None, None),
            element_size=(None, None),
            margins=(None, None),
            element_padding=(None, None),
            auto_size_text=None,
            auto_size_buttons=None,
            font=None,
            border_width=None,
            slider_border_width=None,
            slider_relief=None,
            slider_orientation=None,
            autoclose_time=None,
            message_box_line_width=None,
            progress_meter_border_depth=None,
            progress_meter_style=None,
            progress_meter_relief=None,
            progress_meter_color=None,
            progress_meter_size=None,
            text_justification=None,
            background_color=None,
            element_background_color=None,
            text_element_background_color=None,
            input_elements_background_color=None,
            scrollbar_color=None, text_color=None,
            debug_win_size=(None, None),
            window_location=(None, None)
        )

        # define main window layout
        self.layout = [
            # absobrber info
            [sg.Text(Κωδικός συλλέκτη:,
             size=(14, 1),
             auto_size_text=False,
             justification='right')],
            [sg.InputText(do_not_clear=True, size=(30, 1),
                          enable_events=True, key='absorber_info')
                             ],

            # is horizontal absorber
            [sg.Checkbox(Οριζόντιος, default=False,
             enable_events=True,
             key='is_horizontal')],

            # panel/strips section
            [sg.Frame(Χαρακτηριστικά συλλέκτη  + '(mm)', layout=[
                # header diameter
                [sg.Text(Φ κολεκτέρ, size=(FRAME_LCOL_WIDTH, 1),
                 auto_size_text=False,
                 justification='right'),
                 sg.InputText('22', do_not_clear=True, size=(TB_WID
                    , 1),
                 enable_events=True,
                 key='header_diameter')],
```



```

                # head to head distance
                [sg.Text(Κέντρο-κέντρο, size=(FRAME_LCOL_WIDTH, 1),
                 auto_size_text=False,
                 justification='right'),
                 sg.InputText(do_not_clear=True, size=(TB_WID, 1),
                 enable_events=True,
                 key='head_to_head')],

                # free header exit length
                [sg.Text(Μήκος ελεύθερης  + '\n' + εξόδου κολεκτέρ,
                 size=(FRAME_LCOL_WIDTH, 2),
                 auto_size_text=False,
                 justification='right'),
                 sg.InputText(do_not_clear=True, size=(TB_WID, 1),
                 enable_events=True,
                 key='header_exit_length')],

                # panel width
                [sg.Text(Πλάτος φύλλου,
                 size=(FRAME_LCOL_WIDTH, 1),
                 auto_size_text=False,
                 justification='right'),
                 sg.InputText(do_not_clear=True, size=(TB_WID, 1),
                 enable_events=True,
                 key='panel_width')],

                # panel width
                [sg.Text(Μήκος φύλλου,
                 size=(FRAME_LCOL_WIDTH, 1),
                 auto_size_text=False,
                 justification='right'),
                 sg.InputText(do_not_clear=True, size=(TB_WID, 1),
                 enable_events=True,
                 key='panel_length')]
             ])],

            # error box
            [sg.Text('', text_color='red', size=(27, 3),
                     key='error_box')],

            # buttons
            [sg.Text(' ' * 3),
             sg.Button(Αποστολή, size=(9, 1), key='send_button'),
             sg.Button(Αποθήκευση, size=(9, 1), key='save_button')],
            [sg.Text(' ' * 3),
             sg.Button('Offsets', size=(9, 1), key='offsets_button'
                ),
             sg.Button(Φόρτωση, size=(9, 1), key='load_button')]
        ]

        # define machine offsets window layout
        self.offsets_layout = [
            [sg.Frame(Πλέγμα  + 'X', layout=[
                [sg.Text(Μοτέρ 1 (mm), size=(OFFSETS_LCOL_WIDTH, 1),
                 auto_size_text=False,
                 justification='right'),
                 sg.InputText('', do_not_clear=True, size=(6, 1),
                 enable_events=True,
                 key='motor_1')],
                [sg.Text(Μοτέρ 2 (mm), size=(OFFSETS_LCOL_WIDTH, 1),
                 auto_size_text=False,
                 justification='right'),
                 sg.InputText('', do_not_clear=True, size=(6, 1),
```

```
                    enable_events=True,
                    key='motor_2')]
            ])],
            [sg.Frame(Πλέγμα  + 'Y', layout=[
                [sg.Text(Μοτέρ 3 (mm), size=(OFFSETS_LCOL_WIDTH, 1),
                    auto_size_text=False,
                    justification='right'),
                    sg.InputText('', do_not_clear=True, size=(6, 1),
                    enable_events=True,
                    key='motor_3')],
                [sg.Text(Μοτέρ 4 (mm), size=(OFFSETS_LCOL_WIDTH, 1),
                    auto_size_text=False,
                    justification='right'),
                    sg.InputText('', do_not_clear=True, size=(6, 1),
                    enable_events=True,
                    key='motor_4')]
            ])],
            [sg.Frame(Φύλλο X, layout=[
                [sg.Text(Μοτέρ 5 (deg), size=(OFFSETS_LCOL_WIDTH, 1),
                    auto_size_text=False,
                    justification='right'),
                    sg.InputText('', do_not_clear=True, size=(6, 1),
                    enable_events=True,
                    key='motor_5')],
                [sg.Text(Μοτέρ 6 (deg), size=(OFFSETS_LCOL_WIDTH, 1),
                    auto_size_text=False,
                    justification='right'),
                    sg.InputText('', do_not_clear=True, size=(6, 1),
                    enable_events=True,
                    key='motor_6')]
            ])],
            [sg.Frame(Φύλλο Y, layout=[
                [sg.Text(Μοτέρ 7 (deg), size=(OFFSETS_LCOL_WIDTH, 1),
                    auto_size_text=False,
                    justification='right'),
                    sg.InputText('', do_not_clear=True, size=(6, 1),
                    enable_events=True,
                    key='motor_7')],
                [sg.Text(Μοτέρ 8 (deg), size=(OFFSETS_LCOL_WIDTH, 1),
                    auto_size_text=False,
                    justification='right'),
                    sg.InputText('', do_not_clear=True, size=(6, 1),
                    enable_events=True,
                    key='motor_8')]
            ])],

            # error box
            # [sg.Text('', text_color='red', size=(20, 3),
            #          key='offsets_error_box')],

            # buttons
            [sg.Button('OK', size=(10, 1),
                       key='OK_button'),
             sg.Button(Επαναφορά, size=(10, 1),
                       key='reset_button')]
        ]

        # configuration instance
        self.configuration = Configuration()

        # file manager instance
        self.file_manager = FileManager()

    def is_positive_float(self, string):
```



```
        try:
            if float(string) > 0.0:
                return True
            else:
                return False
        except ValueError:
            return False

    def is_float(self, string):
        try:
            a = float(string)
        except ValueError:
            return False
        else:
            return True

    def check_for_errors(self, values):
        # check for positive floats
        try:
            for key in values.keys():
                # skip absorber info
                if key in ('absorber_info'):
                    continue

                if type(values[key]) == str:
                    if not self.is_positive_float(values[key]):
                        return Σφάλμα:  επιτρέπονται μόνο θετικοί αριθμοί
        except AttributeError:
            return Σφαλμα:  μη έγκυρη τιμή

        size_error = False
        dimension = ''
        comparison = ''

        # check head-to-head+header diameter and fixture length or
           width
        if not values['is_horizontal']:
            effective_length = (values['head_to_head'] +
                                values['header_diameter'])
            effective_width = (values['panel_width'] +
                               2*values['header_exit_length'])
        else:
            effective_length = (values['panel_length'] +
                                2*values['header_exit_length'])
            effective_width = (values['head_to_head'] +
                               values['header_diameter'])

        if (effective_width > self.configuration.MAX_WIDTH):
            size_error = True
            dimension = πλάτος
            comparison = μεγαλύτερο
        elif (effective_width < self.configuration.MIN_WIDTH):
            size_error = True
            dimension = πλάτος
            comparison = μικρότερο

        if (effective_length > self.configuration.MAX_LENGTH):
            size_error = True
            dimension = μήκος
            comparison = μεγαλύτερο
        elif (effective_length < self.configuration.MIN_LENGTH):
            size_error = True
            dimension = μήκος
            comparison = μικρότερο
```

```
297
298         if size_error:
299             return (Σφάλμα: το  + dimension +  του συλλέκτη είναι
300                     + comparison +  από αυτό που εξυπηρετεί η διάταξη.)
301
302         return ''
303
304     def check_offsets_for_errors(self, values):
305         # check for positive floats
306         try:
307             for key in values.keys():
308                 if type(values[key]) == str:
309                     if not self.is_float(values[key]):
310                         return (Σφάλμα: επιτρέπονται μόνο θετικοί
311                                 αριθμοί  + values[key])
312         except AttributeError:
313             return Σφαλμα: μη έγκυρη τιμή
314
315         return ''
316
317     def show_error(self, error):
318         self.window.Element('error_box').Update(error)
319
320     def show_offsets_error(self, error):
321         # self.offsets_window.Element('offsets_error_box').Update(
              error)
322         pass
323
324     def export_configuration_to_file(self, values, file_path):
325         with open(file_path, 'w', encoding='utf-8') as the_file:
326             for key in values.keys():
327                 # write separators
328                 if key == 'header_diameter':
329                     the_file.write(
330                             '#' +  Διάμετρος κολεκτέρ  + '(mm)\n'
331                     )
332                 elif key == 'head_to_head':
333                     the_file.write(
334                         '#' +  Απόσταση κέντρο-κέντρο  + '(mm)\n'
335                     )
336                 elif key == 'header_exit_length':
337                     the_file.write(
338                         '#' + Μήκος ελεύθερης εξόδου  + 'header (mm)\n'
339                     )
340                 elif key == 'panel_width':
341                     the_file.write(
342                         '#' +  Πλάτος φύλλου  + '(mm)\n'
343                     )
344                 elif key == 'panel_lenth':
345                     the_file.write(
346                         '#' +  Μήκος φύλλου  + '(mm)\n'
347                     )
348                 the_file.write(key + '=' + str(values[key]) + '\n\n
                        ')
349
350         self.show_error(Οι ρυθμίσεις αποθηκεύτηκαν στο
351                         αρχείο: + '\n' + self.output_file)
352
353     def import_configuration_from_file(self, values, file_path):
354         if path.isfile(file_path):
355             with open(file_path, encoding='utf-8') as the_file:
356                 for line in the_file:
357                     # skip comment lines
358                     if line.startswith('#'):
```



```
359                         continue
360                     # store appropriate
361                     for key in values.keys():
362                         if line.startswith(key + '='):
363                             values[key] = line.strip('\n').split('=
                                    ')[1]
364                             if key == 'is_horizontal':
365                                 values[key] = values[key] == 'true'
366                             self.window.Element(key).Update(values[
                                    key])
367
368         self.show_error(Οι ρυθμίσεις φορτώθηκαν από το αρχείο: + '\n' +
369                         file_path)
370
371     def export_offsets_to_file(self):
372         with open(self.OFFSETS_FILE, 'w', encoding='utf-8') as
                the_file:
373             # write separators
374             for key in self.offsets.keys():
375                 the_file.write(key+'=' + str(self.offsets[key]) + '
                        \n')
376
377     def import_offsets_from_file(self):
378         # reset imported offsets to defaults
379         imported_offsets = self.default_offsets.copy()
380
381         # if custom offsets file is present in current directory
                then
382         # import these offsets
383         if path.isfile(self.OFFSETS_FILE):
384             with open(self.OFFSETS_FILE, encoding='utf-8') as
                    the_file:
385                 for line in the_file:
386                     # skip comment lines
387                     if line.startswith('#'):
388                         continue
389                     # store appropriate
390                     for key in self.offsets.keys():
391                         if line.startswith(key + '='):
392                             imported_offsets[key] = float(
393                                     line.strip('\n').split('=')[1]
394                             )
395
396             self.show_offsets_error(Τα +'offsets'+φορτώθηκαν από το
397                                     αρχείο: + '\n' + self.
                                        OFFSETS_FILE)
398
399         return imported_offsets
400
401     def fill_offsets_window(self):
402         imported_offsets = self.import_offsets_from_file()
403
404         for key in self.offsets.keys():
405             self.offsets_window.Element(key).Update(
                    imported_offsets[key])
406
407     def run_offsets_window(self):
408         # draw offsets window
409         self.offsets_window = (sg.Window('Offsets',
410                                keep_on_top=True)
411                                .Layout(self.offsets_layout))
412         self.offsets_window.Finalize()
413
414         # initialize offsets
```

```
        self.fill_offsets_window()

        # offsets window event Loop
        while True:
            # get the event and current values
            event, values = self.offsets_window.Read()

            # user closes the window
            if event is None:
                break

            # format values
            for key, val in values.items():
                if self.is_positive_float(val):
                    values[key] = float(val)

            # check for errors in any event except 'reset', 'X', or
                'OK'
            if event not in (None, 'reset_button'):
                self.show_offsets_error('')
                error_found = self.check_offsets_for_errors(values)
                if error_found != '':
                    self.show_offsets_error(error_found)
                    print(error_found)
                    continue

            # user presses 'OK button'
            if event == 'OK_button':
                self.export_offsets_to_file()

                self.show_offsets_error(
                        (Τα +'offsets'+ αποθηκεύτηκαν στο αρχείο: +
                        '\n' + self.OFFSETS_FILE)
                )

                break

            # user presses 'reset' button
            if event == 'reset_button':
                self.fill_offsets_window()

            # if user modifies a value then store it
            if event in self.offsets.keys():
                self.offsets[event] = values[event]

            # DEBUG
            # print('\noffsets event: ')
            # pprint(event)
            # print('\noffsets values: ')
            # pprint(values)

        self.offsets_window.Close()

    def run(self):
        # draw main window
        self.window = sg.Window(Ρύθμιση Διάταξης).Layout(self.layout)

        # main window vent Loop
        while True:
            # get the event and current values
            event, values = self.window.Read()

            # set offsets
            self.offsets = self.import_offsets_from_file()
```



```

        # user closes the window
        if event is None:
            break

        # format values
        for key, val in values.items():
            if key not in ('absorber_info', 'is_horizontal'):
                if self.is_positive_float(val):
                    values[key] = float(val)

        # check for errors in any event except 'reset', 'X', or
            'OK'
        if event not in (None, 'load_button', 'offsets_button',
                         'save_button', 'absorber_info'):
            self.show_error('')
            error_found = self.check_for_errors(values)
            if error_found != '':
                self.show_error(error_found)
                continue

        # user presses 'save button'
        if event == 'save_button':
            self.show_error('')
            error_found = self.check_for_errors(values)
            if error_found != '':
                self.show_error(error_found)
                continue

            # disable buttons
            self.window.Element('save_button').Update(disabled=
                True)
            self.window.Element('send_button').Update(disabled=
                True)
            self.window.Element('load_button').Update(disabled=
                True)
            (self.window.Element('offsets_button')
             .Update(disabled=True))

            # prompt user to supply file name
            self.output_file = sg.PopupGetFile(
                title=Αποθήκευση ρυθμήσεων,
                message=(Παρακαλώ εισάγετε την τοποθεσία του
                         παραγόμενου αρχείου),
                save_as=True,
                default_path=(self.default_output_folder +
                              self.guessed_name + '.dat'),
                default_extension='dat',
                file_types=((Αρχείο ρυθμίσεων, '*.dat'),)
            )

            # if a name is defined then proceed to save
            if self.output_file:
                # ensure correct file extension
                if not self.output_file.endswith('.dat'):
                    self.output_file += '.dat'

                # clean up save directory
                files_were_moved = self.file_manager.
                   prepare_dir(
                    path.basename(self.output_file),
                    path.dirname(self.output_file),
                    self.OLDS_DIR)
```

```
            if files_were_moved:
                pass

            # create configuration DAT
            self.export_configuration_to_file(
                    values, self.output_file
            )

        # re-enable buttons
        self.window.Element('save_button').Update(disabled=
            False)
        self.window.Element('send_button').Update(disabled=
            False)
        self.window.Element('load_button').Update(disabled=
            False)
        (self.window.Element('offsets_button')
         .Update(disabled=False))

    # user presses 'load button'
    if event == 'load_button':
        # disable buttons
        self.window.Element('save_button').Update(disabled=
            True)
        self.window.Element('send_button').Update(disabled=
            True)
        self.window.Element('load_button').Update(disabled=
            True)
        (self.window.Element('offsets_button')
         .Update(disabled=True))

        # prompt user to supply file name
        self.input_file = sg.PopupGetFile(
            title=Φόρτωση ρυθμήσεων,
            message=(Παρακαλώ εισάγετε την τοποθεσία του
                     αρχείου προς φόρτωση),
            save_as=False,
            default_path=(self.default_output_folder),
            default_extension='dat',
            file_types=((Αρχείο ρυθμίσεων, '*.dat'),)
        )

        # if a name is defined then proceed to load
        if self.input_file:
            # ensure correct file extension
            if self.input_file.endswith('.dat'):
                self.import_configuration_from_file(
                    values,
                    self.input_file
                )
            else:
                self.show_error(Μη συμβατό αρχείο προς φόρτωση.)

        # re-enable buttons
        self.window.Element('save_button').Update(disabled=
            False)
        self.window.Element('send_button').Update(disabled=
            False)
        self.window.Element('load_button').Update(disabled=
            False)
        (self.window.Element('offsets_button')
         .Update(disabled=False))

    # user presses 'offsets' button
    if event == 'offsets_button':
```



```
        # disable buttons
        self.window.Element('save_button').Update(disabled=
            True)
        self.window.Element('send_button').Update(disabled=
            True)
        self.window.Element('load_button').Update(disabled=
            True)
        (self.window.Element('offsets_button')
         .Update(disabled=True))

        # open offsets window
        self.run_offsets_window()

        # re-enable buttons
        self.window.Element('save_button').Update(disabled=
            False)
        self.window.Element('send_button').Update(disabled=
            False)
        self.window.Element('load_button').Update(disabled=
            False)
        (self.window.Element('offsets_button')
         .Update(disabled=False))

    # user presses 'send button'
    if event == 'send_button':
        # disable buttons
        self.window.Element('save_button').Update(disabled=
            True)
        self.window.Element('send_button').Update(disabled=
            True)
        self.window.Element('load_button').Update(disabled=
            True)
        (self.window.Element('offsets_button')
         .Update(disabled=True))

        # make configuration
        self.configuration.make(values, self.offsets,
                                self.CONFIG_FILE)

        # re-enable buttons
        self.window.Element('save_button').Update(disabled=
            False)
        self.window.Element('send_button').Update(disabled=
            False)
        self.window.Element('load_button').Update(disabled=
            False)
        (self.window.Element('offsets_button')
         .Update(disabled=False))

    # set possible file name and set panel material
    if event == 'absorber_info':
        # format name
        guessed_name = (values['absorber_info']
                        .split('\n')[0]
                        .strip(',|.| |\t'))

        # format guessed name and drawing number
        for char in ['/', '\\', '|', '*']:
            guessed_name = guessed_name.replace(char, '-')

        for char in ['\t', ':', '?', '<', '>']:
            guessed_name = guessed_name.replace(char, '')

        if guessed_name != '':
```

```
644                     self.guessed_name = guessed_name
645
646             # DEBUG
647             # print('\nevent: ')
648             # pprint(event)
649             # print('\nvalues: ')
650             # pprint(values)
651
652         self.window.Close()
```

configuration_utils.py

```
 1 from numpy import array, pi, arcsin, floor
 2 # from numpy import add as np_add, array
 3 # from pprint import pprint
 4
 5
 6 class Configuration:
 7     def __init__(self):
 8         self.MAX_LENGTH = 2220  # mm
 9         self.MIN_LENGTH = 1120  # mm
10         self.MAX_WIDTH = 1250  # mm
11         self.MIN_WIDTH = 780  # mm
12         self.RIGHT_CENTER_DIST = 160  # mm
13         self.BOTTOM_CENTER_DIST = 160  # mm
14         self.START_ANGLE = 33  # deg
15         self.PANEL_STOP_LENGTH = 240  # mm
16         self.STOP_TIP_RADIUS = 12.5  # mm
17         self.STEP_ANGLE = 1.8  # degrees
18         SCREW_STARTS = 4  # no units
19         SCREW_PITCH = 2  # mm
20         self.SCREW_LEAD = SCREW_STARTS*SCREW_PITCH  # mm
21         self.REDUCER_RATIO = 1  # no units
22
23     def absorber_dimensions(self, values):
24         if not values['is_horizontal']:
25             length = values['head_to_head'] + values['
                   header_diameter']
26             width = values['panel_width'] + 2*values['
                   header_exit_length']
27         else:
28             length = values['panel_length']+2*values['
                   header_exit_length']
29             width = values['head_to_head'] + values['
                   header_diameter']
30
31         dimensions = {
32             'grid_length': length,
33             'grid_width': width,
34             'panel_length': values['panel_length'],
35             'panel_width': values['panel_width'],
36             'header_exit_length': values['header_exit_length'],
37             'is_horizontal': values['is_horizontal']
38         }
39
40         return dimensions
41
42     def vert_bar_steps(self, dimensions, offsets):
43         # calculate vertical bar's steppers' steps
44         distance = motor_rots = motor_steps = array([0, 0])
45
46         distance = self.MAX_WIDTH + offsets - dimensions['
               grid_width']
47         motor_rots = distance/self.SCREW_LEAD  # calculate
```



```
           rotations
48         motor_steps = floor(motor_rots*360/self.STEP_ANGLE)  #
               steps
49         return motor_steps.tolist()
50
51     def horiz_bar_steps(self, dimensions, offsets):
52         # calculate horizontal bar's steppers' steps
53         distance = motor_rots = motor_steps = array([0, 0])
54
55         distance = self.MAX_LENGTH + offsets - dimensions['
               grid_length']
56         motor_rots = distance/self.SCREW_LEAD
57         motor_steps = floor(motor_rots*360/self.STEP_ANGLE)
58         return motor_steps.tolist()
59
60     def vert_panel_steps(self, dimensions, offsets):
61         # calculate vertical (right) panel side's steppers' steps
62         motor_angle = motor_steps = array([0, 0])
63
64         if not dimensions['is_horizontal']:
65             distance = (self.RIGHT_CENTER_DIST - self.
                   STOP_TIP_RADIUS +
66                         dimensions['header_exit_length'])
67         else:
68             distance = (self.RIGHT_CENTER_DIST - self.
                   STOP_TIP_RADIUS -
69                         (dimensions['panel_width'] -
70                          dimensions['grid_width'])/2)
71
72         theta = arcsin(distance/self.PANEL_STOP_LENGTH)*360/(2*pi)
73         motor_angle = theta - self.START_ANGLE + offsets  # radians
74         motor_steps = floor(motor_angle/self.STEP_ANGLE)
75         return motor_steps.tolist()
76
77     def horiz_panel_steps(self, dimensions, offsets):
78         # calculate horizontal (bottom) panel side's steppers'
               steps
79         motor_angle = motor_steps = array([0, 0])
80
81         if not dimensions['is_horizontal']:
82             distance = (self.BOTTOM_CENTER_DIST - self.
                   STOP_TIP_RADIUS -
83                         (dimensions['panel_length'] -
84                          dimensions['grid_length'])/2)
85         else:
86             distance = (self.BOTTOM_CENTER_DIST - self.
                   STOP_TIP_RADIUS +
87                         dimensions['header_exit_length'])
88
89         theta = arcsin(distance/self.PANEL_STOP_LENGTH)*360/(2*pi)
90         motor_angle = theta - self.START_ANGLE + offsets  # radians
91         motor_steps = floor(motor_angle/self.STEP_ANGLE)
92         return motor_steps.tolist()
93
94     def stepper_steps(self, dimensions, offsets):
95         steps = []
96         offsets = array(list(offsets.values()))
97         steps.extend(self.vert_bar_steps(dimensions, offsets[0:2]))
98         steps.extend(self.horiz_bar_steps(dimensions, offsets[2:4])
               )
99         steps.extend(self.vert_panel_steps(dimensions, offsets
               [4:6]))
100        steps.extend(self.horiz_panel_steps(dimensions, offsets
               [6:8]))
```

```
101
102         return steps
103
104     def make(self, values, offsets, file_path):
105         # calculate and export steps to file
106         dimensions = self.absorber_dimensions(values)
107         steps = self.stepper_steps(dimensions, offsets)
108
109         with open(file_path, 'w', encoding='utf-8') as the_file:
110             the_file.write('# steppers\' steps (1 step = 1.8 deg)\n
                ')
111             for i in range(len(steps)):
112                 the_file.write('motor_'+str(i)+': '+str(steps[i])+'
                    \n')
```

file_utils.py

```
1  from os import path, listdir, mkdir, rename
2  # from pprint import pprint
3
4
5  class FileManager:
6      def __init__(self):
7          self.output_file_name = 'name.txt'
8          self.dir_path = '~/Documents'
9          self.OLDS_DIR_NAME = 'old'
10
11     def current_drawing(self, output_file_name, dir_path):
12         cur_drawing = []
13
14         if output_file_name == '':
15             return cur_drawing
16
17         for file_name in listdir(dir_path):
18             if file_name == output_file_name:
19                 cur_drawing = True
20                 break
21
22         return cur_drawing
23
24     def move_file(self, file_name, old_dir, new_dir):
25         # determine appropriate file name
26         new_file_name = file_name
27         count = 1
28         name, extension = path.splitext(file_name)
29
30         while path.isfile(path.join(new_dir, new_file_name)):
31             count += 1
32             new_file_name = name + ' (' + str(count) + ')' +
                extension
33
34         # move the file
35         old_file_path = path.join(old_dir, file_name)
36         new_file_path = path.join(new_dir, new_file_name)
37         rename(old_file_path, new_file_path)
38
39     def prepare_dir(self, output_file_name, dir_path, olds_dir_name
       ):
40         # store any existing file names with same drawing number
41         cur_drawing = self.current_drawing(output_file_name,
              dir_path)
42
43         if cur_drawing:
44             # if olds directory doesn't exist, create it
```



```
45             olds_dir_path = dir_path + '\\' + olds_dir_name
46             if not path.isdir(olds_dir_path):
47                 mkdir(olds_dir_path)
48
49             old_drawings_path = (olds_dir_path + '\\' +
50                                  output_file_name.split('.', 1)[0])
51             if not path.isdir(old_drawings_path):
52                 mkdir(old_drawings_path)
53
54             # move existing drawing to olds dir without overwriting
55             self.move_file(output_file_name, dir_path,
                  old_drawings_path)
56
57             files_were_moved = True
58         else:
59             files_were_moved = False
60
61         return files_were_moved
```

# Παράρτημα Γ Κώδικας μικρο-ελεγκτή Arduino

program.ino

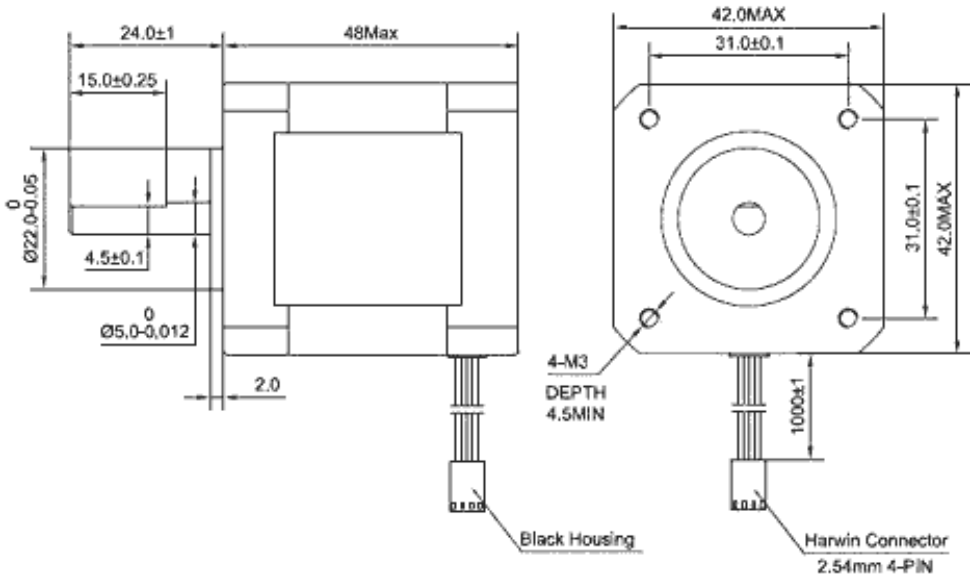
```
#define LENGTH(array) ( sizeof(array)/sizeof(array[0]) )

#define HOME(steppers) { \
    waitForSerial(); \
    shiftArr(LENGTH(steppers), steppers, -1); \
    homeSteppers(LENGTH(steppers), steppers); \
    shiftArr(LENGTH(steppers), steppers, +1); \
}

#define MOVE(steppers) { \
    waitForSerial(); \
    Serial.print("\n-- Insert steps: "); \
    totalSteps = readFromSerial(); \
    Serial.println(totalSteps); \
    shiftArr(LENGTH(steppers), steppers, -1); \
    moveSteppers(LENGTH(steppers), steppers, totalSteps); \
    shiftArr(LENGTH(steppers), steppers, +1); \
}

// steppers
int VERT_BAR_STEPPERS[]   = {1, 2};
int VERT_STOP_STEPPERS[] = {3};

const int STEPS_PER_REVOLUTION = 200;

// pins numbers
const int PUL_PIN[] = {3, 6, 11};
const int DIR_PIN[] = {2, 7, 12};
const int  EN_PIN[] = {4, 8, 13};
const int  SW_PIN[] = {5, 9, 10};

// motion parameters
int   totalSteps = 0;
float rot_speed  = 10; // rpm
float interval   = 1000.0*60.0/(rot_speed*STEPS_PER_REVOLUTION)
   /2.0;

void initializePins() {
    for (int i = 0; i < LENGTH(PUL_PIN); i++) {
        // set and initialize the pins
        pinMode(PUL_PIN[i], OUTPUT);
        pinMode(DIR_PIN[i], OUTPUT);
        pinMode( EN_PIN[i], OUTPUT);
        pinMode( SW_PIN[i], INPUT_PULLUP);

        digitalWrite(DIR_PIN[i], HIGH); // forward rotation
    }
}

void waitForSerial() {
    Serial.println("\n[Press Return key to continue]");
    while (Serial.available()) { Serial.read(); }
    while (!Serial.available()) {}
    //Serial.println(Serial.read()); // print read value
    Serial.println("[Continuing]");
}

void shiftArr(int arrayLength, int array[], int value) {
    for (int i = 0; i < arrayLength; i++) {
        array[i] += value;
    }
}

void SerialPrintSteppers(int stepperCount, int steppers[]) {
    Serial.print("[");
    Serial.print(steppers[0] + 1);

    for (int i = 1; i < stepperCount; i++) {
        Serial.print(", ");
        Serial.print(steppers[i] + 1);
    }

    Serial.print("]");
}

void homeSteppers(int stepperCount, int steppers[]) {
    Serial.print("\n== Homing motor(s) ");
    SerialPrintSteppers(stepperCount, steppers);
    Serial.println(".");

    int finished = 0;

    // reverse rotation for return to home position
    for (int i = 0; i < stepperCount; i++) {
        digitalWrite(DIR_PIN[steppers[i]], HIGH);
    }

    while(1) {
        for (int i = 0; i < stepperCount; i++) {
            if (digitalRead(SW_PIN[steppers[i]]) == HIGH) {
                // limit switch not pressed, keep moving
                moveSteppers(1, &steppers[i], 1);
            } else {
                // change to forward rotation
                Serial.print(" ----- Limit switch on pin ");
                Serial.print(SW_PIN[steppers[i]]);
                Serial.println(" closed.");
            }
        }

        finished = 1;
        for (int i = 0; i < stepperCount; i++) {
            if (digitalRead(SW_PIN[steppers[i]]) == HIGH) {
                finished = 0;
            }
        }

        if (finished == 1) {
            for (int i = 0; i < stepperCount; i++) {
                digitalWrite(DIR_PIN[steppers[i]], LOW);
            }
            break;
        }

        delay(interval);
    }

    Serial.println("\n== Homing finished.");
}

void moveSteppers(int stepperCount, int steppers[], int totalSteps)
    {
    String sign;
    if (digitalRead(DIR_PIN[steppers[0]]) == LOW) {
```

```
        sign = "-";
    } else {
        sign = "+";
    }

    Serial.print("\n-- Rotating motor(s) ");
    SerialPrintSteppers(stepperCount, steppers);
    Serial.print(" by ");
    Serial.print(sign);
    Serial.print(totalSteps);
    Serial.println(" step(s).");

    for (int step = 0; step < totalSteps; step++) {
        for (int i = 0; i < stepperCount; i++) {
            digitalWrite(PUL_PIN[steppers[i]], HIGH);
        }
        delay(interval);

        for (int i = 0; i < stepperCount; i++) {
            digitalWrite(PUL_PIN[steppers[i]], LOW);
        }
        delay(interval);

        // show progress
        Serial.println(
            "step: " + sign + (step + 1) + "/" + sign + totalSteps
        );
    }

    Serial.println("-- Rotating finished.");
}

int readFromSerial() {
    Serial.flush();
    String inputString = "";

    // Read integer value from serial input:
    Serial.read();
    while (1) {
        while (!Serial.available()) {}
        inputString = Serial.readStringUntil('\n');

        if (inputString.toInt() > 0) {
            // convert the incoming byte to a char
            // and add it to the string
            return inputString.toInt();
        } else {
            Serial.println("Please insert a positive integer.");
        }
    }
}

void setup() {
    // start the serial
    Serial.begin(9600);

    // itialize pins
    initializePins();
}

void loop() {
    Serial.flush();

    // show info
```

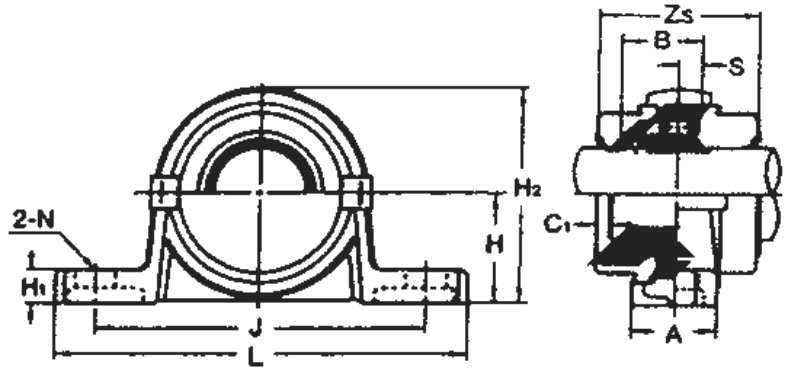


```
    Serial.println("----------------------------------------");
    Serial.println("---------------- START -----------------");
    Serial.println("----------------------------------------");
    Serial.print("Rotational speed = ");
    Serial.print(rot_speed);
    Serial.println(" rpm");
    Serial.print("Interval = ");
    Serial.print(interval);
    Serial.println(" ms");

    // home and move vertical bar's stepper pair
    HOME(VERT_BAR_STEPPERS);
    MOVE(VERT_BAR_STEPPERS);

    // home and move horizontal panel stop's stepper pair
    HOME(VERT_STOP_STEPPERS);
    MOVE(VERT_STOP_STEPPERS);

    Serial.println("----------------------------------------");
    Serial.println("---------------- FINISH ----------------");
    Serial.println("----------------------------------------");

    // wait for keypress
    waitForSerial();
}
```

# Παράρτημα Δ Τεχνικά χαρακτηριστικά στοιχείων

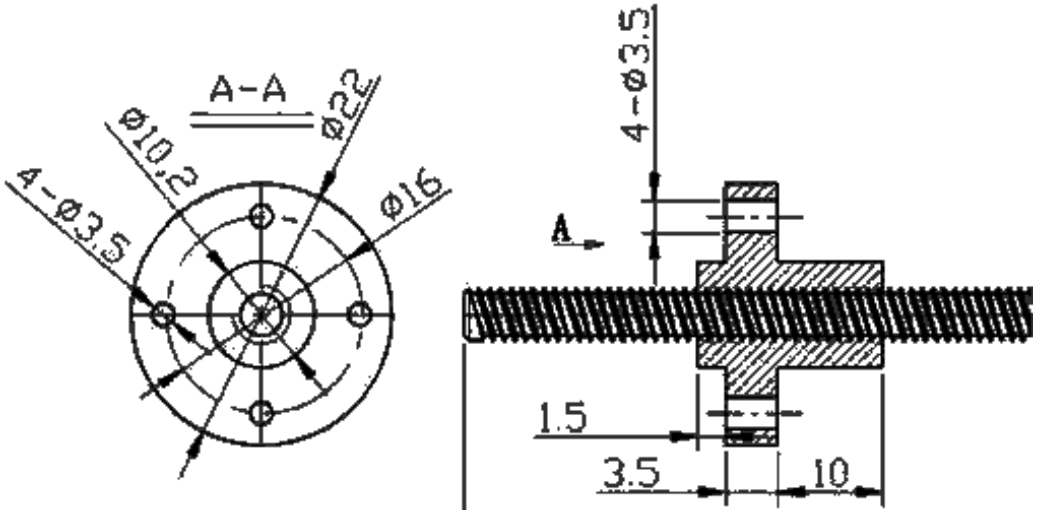



Σχήμα Δ1: Διαστάσεις βηματικού κινητήρα κατηγορίας διαστάσεων NEMA 17.



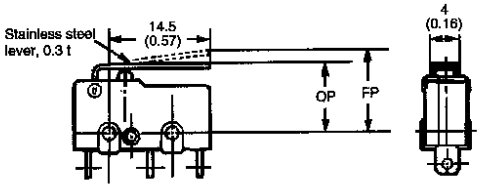

| 带座轴承型号NO. | 轴径 | 外形尺寸Dimensions(mm) | | | | | | | | | | | 螺栓 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | (mm) | H | L | J | A | N | H1 | H2 | Bl | S | Cl | ZS | (mm) |
| KP08 | 8 | 15 | 55 | 42 | 13 | 4.8 | 5 | 29 | 15 | 3.5 | | | M4 |
| KP000 | 10 | 18 | 67 | 53 | 16 | 7 | 6 | 35 | 14 | 4 | 5.6 | 33 | M6 |
| KP001 | 12 | 19 | 71 | 56 | 16 | 7 | 6 | 38 | 15 | 4 | 5 | 33 | M6 |
| KP002 | 15 | 22 | 80 | 63 | 16 | 7 | 7 | 43 | 17 | 4.5 | 4 | 34 | M6 |
| KP003 | 17 | 24 | 85 | 67 | 18 | 7 | 7 | 47 | 18 | 5 | 5.5 | 38 | M6 |
| KP004 | 20 | 28 | 100 | 80 | 20 | 10 | 9 | 55 | 21 | 6 | 7 | 46 | M8 |
| KP005 | 25 | 32 | 112 | 90 | 20 | 10 | 10 | 62 | 23 | 6 | 6 | 47 | M8 |
| KP006 | 30 | 36 | 132 | 106 | 26 | 13 | 11 | 70 | 25 | 6.5 | 6 | 50 | M10 |
| KP007 | 35 | 40 | 150 | 118 | 26 | 13 | 13 | 80 | 28 | 7 | 6 | 54 | M10 |

Σχήμα Δ2: Διαστάσεις των μπλοκ ρουλεμάν του ατέρμονα κοχλία (χρησιμοποιήθηκε το μοντέλο KP08).




Σχήμα Δ3: Διαστάσεις των περικοχλίων του ατέρμονα κοχλία.

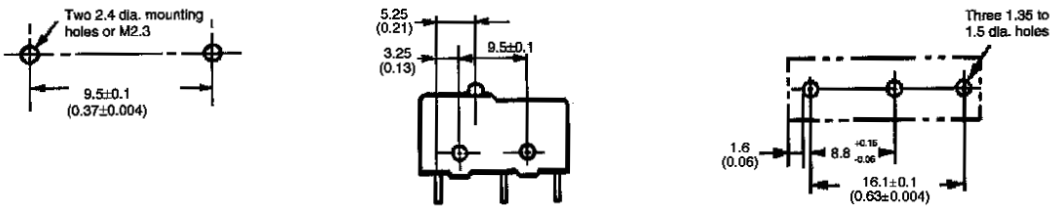
**Hinge lever**
**SS-01GL (-E, -F)**
**SS-5GL (-F)**
**SS-10GL**




| Operating characteristics | Super-low force model SS-01GL-E | Low-force models SS-01GL-F SS-5GL-F | Standard models SS-01GL SS-5GL | Standard model SS-10GL |
|---|---|---|---|---|
| OF max. | 8 g | 16 g | 50 g | 50 g |
| RF min. | 1 g | 2 g | 6 g | 6 g |
| OT min. | 1.2 mm (0.472 in) | 1.2 mm (0.472 in) | 1.2 mm (0.472 in) | 1.0 mm (0.039 in) |
| MD max. | 0.8 mm (0.032 in) | 0.8 mm (0.032 in) | 0.8 mm (0.032 in) | 1.0 mm (0.039 in) |
| FP max. | 13.6 mm (0.535 in) | 13.6 mm (0.535 in) | 13.6 mm (0.535 in) | 13.6 mm (0.535 in) |
| OP | 8.8 ± 0.8 mm (0.346 ± 0.032 in) | 8.8 ± 0.8 mm (0.346 ± 0.032 in) | 8.8 ± 0.8 mm (0.346 ± 0.032 in) | 8.8 ± 0.8 mm (0.346 ± 0.032 in) |

## ■ MOUNTING HOLES

**Standard types**
Mounting holes below are for reference only.




Note: Unless otherwise specified, a tolerance of ± 0.4 mm applies to all dimensions.

Σχήμα Δ4: Διαστάσεις οριακού διακόπτη Omron SS-5GL.

# Παράρτημα Ε Κατασκευαστικά σχέδια τεμαχίων

Ακολουθούν τα κατασκευαστικά σχέδια των τεμαχίων που χρησιμοποιήθηκαν στη διάταξη.