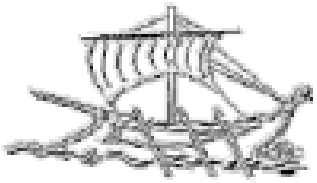


ΑΕΙ ΠΕΙΡΑΙΑ Τ.Τ.

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Σχεδιασμός & Ανάπτυξη Πλατφόρμας Αποκατάστασης Άνω Άκρων Ασθενών

Βασίλης Σ. Λασκαρίδης

Εισηγητής: Δρ. Ιωάννης Έλληνας, Καθηγητής

ΑΘΗΝΑ

ΣΕΠΤΕΜΒΡΙΟΣ 2016

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Σχεδιασμός & Ανάπτυξη Πλατφόρμας Αποκατάστασης Άνω Άκρων Ασθενών

Βασίλης Σ. Λασκαρίδης
Α.Μ. 41794

Εισηγητής:

Δρ. Ιωάννης Έλληνας, Καθηγητής

Εξεταστική Επιτροπή:

Ημερομηνία εξέτασης: 13/09/2016

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Βασίλης Λασκαρίδης του Σωτηρίου, με αριθμό μητρώου 41794 φοιτητής του Τμήματος Μηχανικών Η/Υ Συστημάτων του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των οποίων ποινικών ευθυνών του συγγραφέα, σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή.

Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

ΕΥΧΑΡΙΣΤΙΕΣ

Ολοκληρώνοντας ένα δύσκολο έργο όπως αυτό της Πτυχιακής Εργασίας, θα ήθελα να ευχαριστήσω συγκεκριμένα άτομα, χωρίς τα οποία δε θα την έφερνα εις πέρας.

Η παρούσα πτυχιακή ήταν μια ιδέα του κ. Απόστολου Κατούνα ο οποίος ήθελε να χρηματοδοτήσει και δωρίσει μια τέτοια συσκευή στο Εθνικό Κέντρο Αποκατάστασης λόγω της νοσηλείας συγγενικού του προσώπου σε αυτό. Από τη θέση αυτή θα ήθελα να ευχαριστήσω τον κ. Κατούνα αλλά και τους κ.κ. Κέλλυ Λάππα και Δρ Γιώργο Μαρινάκη από το ΕΚΑ για τις υποδείξεις σχετικά με την κατασκευή του πρωτοτύπου.

Επίσης θέλω να ευχαριστήσω τον κ. Ιωάννη Αμοργίνο, επιστημονικό συνεργάτη του ΑΕΙ Πειραιά, για τις πολύτιμες συμβουλές του σχετικά με την σχεδίαση του μηχανολογικού μέρους της πτυχιακής εργασίας.

Την προσπάθειά μου αυτή υποστήριξε ο επιβλέπων καθηγητής μου, Δρ. Ιωάννης Έλληνας, τον οποίο θα ήθελα να ευχαριστήσω θερμά, για την καθοδήγησή του, τις συμβουλές του και την εμπιστοσύνη που έτρεφε στο πρόσωπό μου, η οποία με εμψύχωνε να συνεχίσω.

Ακόμα θα ήθελα να ευχαριστήσω τον κ. Θοδωρή Ποπώφ, Μηχανικό Βιοϊατρικής Τεχνολογίας του Ωνασείου Καρδιοχειρουργικού Κέντρου για τις πολύτιμες συμβουλές και την βοήθειά του στην συγγραφή της εργασίας μου.

Τέλος, ευχαριστώ τους γονείς μου, οι οποίοι με ενεθάρρυναν και με στήριζαν αδιάλειπτα στην ολοκλήρωση της πτυχιακής μου εργασίας.

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία αφορά ένα όργανο εκγύμνασης άνω άκρων, για την ταχύτατη αποκατάσταση ασθενών με δυσκαμψία ή ατροφία, κ.α. που προκλήθηκε κυρίως από εγκεφαλικό επεισόδιο.

Το όργανο μειώνει την βαρύτητα των άνω άκρων, ώστε να διευκολύνει την κίνησή τους κατά την διάρκεια των ασκήσεων αποκατάστασης:

(α) Κατά την εκγύμναση, παροτρύνει τον ασθενή μέσω ηχητικών μηνυμάτων για την επανάληψη των ασκήσεων,

(β) ενώ ταυτόχρονα προβάλλει το score (μερικό ή συνολικό) στην προσαρτημένη οθόνη, που βρίσκεται δεξιόστροφά του.

Το όργανο εκγύμνασης αποτελείται από ηλεκτρονικά και μηχανικά μέρη.

Τα ηλεκτρονικά μέρη του (software & hardware) καταγράφουν τις μοίρες που κάνει κάθε άρθρωση (ώμου, αγκώνα), υπολογίζοντας τις επιτευχθείσες μοίρες των αρθρώσεων και διατηρούν ένα ιστορικό αρχείο ανά ασθενή, ώστε να χρησιμοποιηθούν τα δεδομένα για περαιτέρω στατιστική ανάλυση από τον φυσικοθεραπευτή ή ιατρό.

Τα μηχανικά μέρη είναι κατασκευασμένα από ανοξείδωτες κοιλοδοκούς, οι διακυμάνσεις των οποίων επιτυγχάνονται με στρόφιγγες, οι οποίες προσαρμόζονται στις εκάστοτε διαστάσεις των ασθενών.

ABSTRACT

The arm has been designed and constructed in order to support the physical restoration of patients with upper limb malfunctions (stiffing, atrophy etc).

The arm reduces the upper limb “gravity” to facilitate the patient’s moves during his exercising program and encourages the patient to repeat the exercises by:

(a) Audible signals, a music tone which is heard when the patient makes a new score and

(b) An LCD screen, which shows the score of the patient.

The arm consists of electronic and mechanical parts.

The electronic parts of the arm (software and hardware) monitor and measure the angular movement of each patient’s joint (arm, elbow) and keep a track record of each patient for further statistic analysis by the supervising physiotherapist or doctor.

The mechanical parts are made of stainless steel sections, which adjust to each other by faucets, in order to suit to the patient.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ:

Ηλεκτρονικά Κυκλώματα, 3D Σχεδίαση

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ:

3D Design, C++, Gravity Eliminated Rehabilitation

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΕΦΑΛΑΙΟ 1 : ΕΙΣΑΓΩΓΗ

1.1 Επίπτωση του εγκεφαλικού επεισοδίου

Το Εγκεφαλικό επεισόδιο είναι η κύρια αιτία μόνιμης αναπηρίας των ενηλίκων σε όλο τον κόσμο. Συχνά το εγκεφαλικό επεισόδιο επηρεάζει τον έλεγχο της κίνησης των άνω άκρων με αποτέλεσμα την δυσκολία εκτέλεσης των δραστηριοτήτων της καθημερινής ζωής. Πολλές ώρες αφιερώνονται στη θεραπεία αποκατάστασης, ώστε να ανακτήθει η λειτουργικότητα των άνω άκρων.

Εγκεφαλικό επεισόδιο είναι μια απώλεια της εγκεφαλικής λειτουργίας λόγω διαταραχής στην παροχή αίματος στον εγκέφαλο. Το εγκεφαλικό επεισόδιο μπορεί να είναι είτε αιμορραγικό ή ισχαιμικό.

Στο αιμορραγικό εγκεφαλικό επεισόδιο, τα αιμοφόρα αγγεία διαρρηγνύονται, με αποτέλεσμα την ενδοκρανιακή συσσώρευση του αίματος, ενώ το ισχαιμικό εγκεφαλικό επεισόδιο είναι μια διαταραχή στην αιμάτωση (απόφραξη) και αντιπροσωπεύει περίπου το 80% όλων των περιστατικών εγκεφαλικού επεισοδίου. Η απόφραξη μιας σημαντικής αρτηρίας μπορεί να προκληθεί από διάφορες διαδικασίες που περιλαμβάνουν αθηροσκλήρωση, θρόμβωση, νόσο των μικρών αγγείων και εμβολή.

Χωρίς αίμα για την παροχή οξυγόνου και θρεπτικών ουσιών, τα κύτταρα του εγκεφάλου σε μια εστιακή περιοχή του εγκεφάλου γρήγορα αρχίζουν να πεθαίνουν. Όταν συμβεί αυτό, οι συνέπειες για το άτομο μπορεί να περιλαμβάνουν κινητικές και αισθητικές διαταραχές, προβλήματα όρασης, δυσκολίες στην ομιλία καθώς και γνωστικές ή συναισθηματικές δυσκολίες.

Ανάλογα με τη θέση και τη σοβαρότητα του εγκεφαλικού εμφράγματος, ο ασθενής μπορεί να είναι παρετικός ή ακόμη παράλυτος, είτε από την αριστερή ή δεξιά πλευρά του σώματος, επειδή η ετερόπλευρη πληγείσα περιοχή στον εγκέφαλο δεν μπορεί πλέον να λειτουργήσει.

Κάθε χρόνο προσβάλλονται 16.300.000 άνθρωποι από εγκεφαλικό επεισόδιο, σε όλο τον κόσμο. Είναι η δεύτερη κύρια αιτία θανάτου και η

κύρια αιτία μόνιμης αναπηρίας σε ενήλικες. Από τους ασθενείς που έχουν βιώσει ένα εγκεφαλικό επεισόδιο υπάρχουν περίπου 64,5 εκατομμύρια επιζήσαντες, που ζουν με διαφορετικά επίπεδα αναπηρίας και χρειάζονται βοήθεια για τις δραστηριότητες της καθημερινής ζωής.

Η ζήτηση είναι ακόμη πιο επιτακτική, αν συνυπάρχουν πολλές νευρολογικές βλάβες εκτός από το εγκεφαλικό επεισόδιο.

1.2 Δυνατότητα για ανάκτηση

Λόγω της πλαστικότητας του εγκεφάλου, είναι δυνατόν να επιτευχθεί σημαντική ανάκαμψη των λειτουργιών αυτού. Η έννοια της πλαστικότητας του εγκεφάλου διατείνεται ότι οι νευρωνικές συνδέσεις του φλοιού μπορεί να αναδιαμορφωθούν, με βάση τις εμπειρίες μας σε όλη μας τις ζωή.

Η ικανότητα του εγκεφάλου να προσαρμοστεί στις συγκεκριμένες συνθήκες, με φυσική αναδιοργάνωση ορισμένων μερών του εαυτού του είναι μια βασική έννοια στην αποκατάσταση μετά από εγκεφαλικό επεισόδιο. Οι μηχανισμοί της αναδιοργάνωσης είναι αβέβαιες, ωστόσο υπάρχουν ενδείξεις ότι η αύξηση της δραστηριότητας των νευρώνων οδηγεί στην μεταβολή του αριθμού των συναπτικών συνδέσεων και σε αυξημένη δενδριτική διακλάδωση.

Πολλές ώρες δαπανώνται για την αποκατάσταση των επιζήσαντων ενός εγκεφαλικού επεισοδίου, προκειμένου να βελτιώσουν τις λειτουργίες τους, κάτι που θα τους επιτρέψει να ζήσουν περισσότερο ανεξάρτητα. Ένα μεγάλο μέρος της αποκατάστασης της βλάβης από το εγκεφαλικό επεισόδιο προϋποθέτει τη βελτίωση της κινητικής λειτουργίας. Περίπου το 80% των ασθενών εγκεφαλικού επεισοδίου έχουν κινητικά ελλείμματα, τα οποία μπορεί να οδηγήσουν σε σοβαρή αναπηρία.

Συχνά, τα άνω άκρα είναι αυτά που έχουν τις περισσότερες σοβαρές βλάβες, επειδή η αρτηρία του εγκεφάλου που είναι υπεύθυνη για την ροή του αίματος, σε περιοχές που ελέγχουν τα άνω άκρα είναι πιο συχνά εμπλεκόμενη, σε ένα εγκεφαλικό επεισόδιο.

ΚΕΦΑΛΑΙΟ 2 : ΣΤΟΧΟΙ

2.1 Στόχοι οργάνου

Ο κύριος στόχος της εργασίας αυτής είναι να σχεδιάσει και να κατασκευάσει ένα όργανο εκγύμνασης για την αποκατάσταση ασθενών επιζήσαντες εγκεφαλικού επεισοδίου, που να μπορεί να χρησιμοποιηθεί στο Εθνικό Κέντρο Αποκατάστασης Χασιάς.

Στην επίσκεψή μου στο Κέντρο, οι υπεύθυνοι φυσικοθεραπευτές παρουσίασαν την μέχρι τώρα μεθοδολογία ασκήσεων αποκατάστασης των ασθενών τους, σε ένα από τα εργαστήρια, που ήθελαν να εκσυγχρονίσουν. Η διαδικασία εκτέλεσης της άσκησης αφορά τα εξής στάδια :

1^{ο} στάδιο: Ο ασθενής κάθεται σε μια καρέκλα πάνω από την οποία κρέμονται 4 σχοινιά, 2 για κάθε χέρι.

2^{ο} στάδιο: Ο ασθενής φοράει τα δύο σχοινιά στο ένα χέρι, ένα στο μπράτσο και ένα λίγο πιο κάτω από τον αγκώνα, με αποτέλεσμα να εξαλύπτεται η βαρύτητα από τα χέρια του ασθενή. Η καρέκλα βρίσκεται σε προκαθορισμένη απόσταση μπροστά από ένα καθρέφτη που έχει κολλημένες κάθετες λωρίδες σε μια συγκεκριμένη απόσταση, η μία από την άλλη.

3^{ο} στάδιο: Ο χρήστης ξεκινάει την εκγύμναση έχοντας ως στόχο να φτάσει κάποια από τις λωρίδες προς δεξιά ή την αριστερή πλευρά. Η πρόοδος φαίνεται από το πόσες περισσότερες λωρίδες μπορούσε να περάσει το χέρι του ασθενή, σε σχέση με την προηγούμενη προπόνηση.

Στις συζητήσεις που είχα με τους φυσικοθεραπευτές του Κέντρου εδραιώθηκε η απόφασή μου για την δημιουργία ενός οργάνου αποκατάστασης ασθενών, με στόχο την ταχύτερη ανάρρωση τους από εγκεφαλικά επεισόδια, με ήπια έως μέτρια ανεπάρκεια των άνω άκρων λόγω ημιπάρεσης.

Αυτοί οι ασθενείς θα πρέπει να έχουν την ικανότητα να κινούν οικειοθελώς το/τα προσβεβλημένο/α άκρο/α, έτσι ώστε η κίνηση να μπορεί να πραγματοποιηθεί χωρίς φυσική βοήθεια από τον φυσικοθεραπευτή.

Το όργανο εκγύμνασης θα επιτρέπει στον ασθενή να εκτελέσει ασκήσεις σε οριζόντιο επίπεδο χωρίς περιορισμούς (αντιδράσεις δυνάμεων), έτσι ώστε να μπορεί να φτάσει όλο το εύρος της κίνησης.

Η πλατφόρμα της θεραπείας θα είναι σε θέση να υποστηρίζει το χέρι του ατόμου, ενάντια στις δυνάμεις της βαρύτητας. Με την μέθοδο της εξάλειψης της βαρύτητας, θα είναι πιο εύκολη η επανάληψη των ασκήσεων.

Η αξιολόγηση της απόδοσης του ασθενή γίνεται μέσω ποσοτικών ενδείξεων (γωνίες των αρθρώσεων). Τα δεδομένα αυτά παρέχονται στον ασθενή, μέσω μιας οθόνης, η οποία συνδέεται με έναν μικροϋπολογιστή.

Σε κάθε νέα βελτίωση του ασθενή, δηλαδή υπέρβαση του προηγούμενου μεγίστου ορίου επαναλήψεων της άσκησης, θα ακούγεται ένα ηχητικό σήμα επιβράβευσης. Αυτή η πληροφορία μπορεί να αποθηκευτεί (ηλεκτρονικά), ώστε να αξιολογηθεί από τον φυσικοθεραπευτή, προκειμένου να τεθούν μελλοντικοί στόχοι της θεραπείας.

Όλα τα μηχανικά, ηλεκτρονικά μέρη καθώς και το λογισμικό σχεδιάστηκαν, ώστε να είναι ασφαλής η χρήση για τον ασθενή.

2.2 Σκοπός της διατριβής

Ο πρωταρχικός σκοπός αυτής της εργασίας είναι η δημιουργία ενός οργάνου εκγύμνασης, που να μπορεί να χρησιμοποιηθεί από ασθενείς εγκεφαλικών επεισοδίων και να παρέχει λεπτομερή ενημέρωση σχετικά με την πρόοδο του ασθενή.

Η δημιουργία «Universal» φυσικοθεραπευτικό μηχάνημα που αφορά όλες τις ανάγκες των ασθενών είναι ουτοπία**.** Πλέον, οι κατασκευαστές προσανατολίζονται στην κατασκευή μηχανημάτων που υποστηρίζουν

την άσκηση ορισμένων μελών του σώματος ή κατασκευάζουν μηχανήματα για συγκεκριμένες παθήσεις.

Ο όρος πλατφόρμα δεν χρησιμοποιείται τυχαία. Δημιουργήθηκε ένα όργανο εκγύμνασης, στο οποίο μπορούμε να χτίσουμε διάφορες φυσικοθεραπευτικές ασκήσεις (εφαρμογές), δηλαδή είναι η βάση πάνω στην οποία μπορούμε να προσθέσουμε καινούργιες «modules», ώστε να επεκτείνουμε τις λειτουργίες του μηχανήματος ή να μεταβάλλουμε τις ήδη υπάρχοντες σύμφωνα με τις ανάγκες των ασθενών, που θέλουμε να προπονούμε.

Είναι εντυπωσιακό το γεγονός ότι σε αυτή την “zero” version της πλατφόρμας, με μια απλή στροφή κατά 90 μοίρες του βασικού εξαρτήματος της διάταξης του όργανου εκγύμνασης και με ελάχιστη αλλαγή στον πηγαίο κώδικα του μικροελεγκτή, η πλατφόρμα μετατρέπεται ριζικά από ασκήσεις σε οριζόντιο επίπεδο, σε ασκήσεις σε κατακόρυφο επίπεδο.

Σε αυτή την αρχική φάση ανάπτυξης του όργανου εκγύμνασης, μπορούμε να πούμε ότι ανήκει στην κατηγορία των παθητικών φυσικοθεραπευτικών συσκευών, δηλαδή απευθύνεται σε ασθενείς με ήπια έως μέτρια ανεπάρκεια των άνω άκρων, οι οποίοι δεν χρειάζονται υποβοήθηση μηχανική, ή υποβοήθηση από τον φυσικοθεραπευτή τους.

Σημειωτέον ότι είναι δυνατή η σταδιακή επέκταση του οργάνου συμπλήρωνοντάς το με νέους μηχανισμούς, π.χ. μπορούμε να μετατρέψουμε το όργανο παθητικής εκγύμνασης σε όργανο εκγύμνασης με υποβοηθούσα διάταξη, προσθέτοντας ένα μοτέρ για κάθε κλείδωση.

2.3 Μέθοδοι αποθεραπείας

ΡΟΜΠΟΤΙΚΟ ΣΥΣΤΗΜΑ ΥΠΟΒΟΗΘΗΣΗΣ ΚΑΤΩ ΑΚΡΩΝ
REO AMBULATOR

Η Θεραπεία Reo™ προσφέρει Λύση για Όλο το Σώμα για Θεραπεία Ρομποτικής Υποβοήθησης. Η θεραπεία αυτή συνδυάζει για πρώτη φορά, καινοτομικές πλατφόρμες ρομποτικής υποβοήθησης για θεραπεία χεριών και ποδιών, έτσι ώστε να παρέχει μεγαλύτερη δυνατότητα αποκατάστασης ασθενών και καλύτερη επιχειρηματική απόδοση. Έχει

σχεδιαστεί για χρήση από φυσιοθεραπευτές και εργοθεραπευτές σε χώρους αποκατάστασης.

Το Reo Ambulator είναι μία καινοτόμος ρομποτική συσκευή εκπαίδευσης βαδίσματος όπου ενσωματώνει το σωματικό βάρος σε υποστηριζόμενο εκπαιδευτικό διάδρομο με προηγμένη ρομποτική τεχνολογία με σκοπό την αποκατάσταση ασθενών που υποφέρουν από ορθοπεδικούς τραυματισμούς ή νευρολογικές δυσλειτουργίες που επηρεάζουν το βάδισμα και τις λειτουργίες της ισορροπίας, τον κινητικό συντονισμό, τη στάση σώματος και την αντοχή.

Το Reo Ambulator διατηρεί με ασφάλεια τον ασθενή σε όρθια θέση, ενώ τα ρομποτικά μέρη υποβοηθούν τα πόδια του ασθενούς κατά το βάδισμα πάνω σε διάδρομο. Προηγμένο λογισμικό ελέγχει το Reo Ambulator μέσω του ενσωματωμένου υπολογιστικού του συστήματος, ενώ αισθητήρες ανιχνεύουν διάφορες λειτουργίες παρακολουθώντας συνεχώς και προσαρμόζοντας την ισχύ και την ταχύτητα ανάλογα με τις απαιτήσεις της φυσιολογίας κάθε ασθενούς.

- Επιτρέπει στους ασθενείς να συμμετέχουν στην κίνηση αλλά παρέχει υπολειπόμενη ισχύ, απαραίτητη για το βάδισμα
- Τα συγχρονισμένα ρομποτικά πόδια επιτρέπουν στους ασθενείς να περπατούν με ασφάλεια κάνοντας χρήση συνήθων προτύπων βαδίσματος
- Επιτρέπει στους θεραπευτές να ρυθμίζουν το ποσοστό βάρους εφαρμόζοντας το ατομικό πλάνο αποκατάστασης κάθε ασθενούς
- Η ταχύτητα βαδίσματος μπορεί να μεταβληθεί σύμφωνα με τις ανοχές του κάθε ασθενούς και να αυξηθεί σταδιακά για να αναπαραχθούν τα κανονικά πρότυπα βαδίσματος

<u>Τα πλεονεκτήματα του Reo Ambulator</u>

Συνδυάζοντας την καινοτόμο και την ενσωματωμένη εκπαιδευτική πλατφόρμα, υποβοηθούμενη από το ρομποτικό μηχανισμό, με την βαθιά κατανόηση για νευρολογική αποκατάσταση, το Reo Ambulator προσφέρει την πιο προηγμένη λύση αποκατάστασης:

- Εντατική και ελεγχόμενη θεραπεία εκπαίδευσης βαδίσματος για ενήλικες και παιδιά με νευρολογικές δυσλειτουργίες και ορθοπεδικούς τραυματισμούς.
- Ακολουθεί δυναμικά τη διαδικασία της αποκατάστασης χρησιμοποιώντας δύο μεθόδους: παθητικά, χρησιμοποιώντας δύο ρομποτικά πόδια που υποβοηθούν ή ελέγχουν την κίνηση και ενεργητικά για πιο προχωρημένα στάδια αποκατάστασης.
- Αυξάνει την επίγνωση του ασθενούς, την αντίληψη, και τον παροτρύνει μέσω ενός μηχανήματος πολλαπλών δυνατοτήτων και γνωστικής εκπαίδευσης χρησιμοποιώντας πλατφόρμα τεχνολογίας VR (Εικονικής Πραγματικότητας).
- Αυξάνει αποδοτικά την θεραπεία της κίνησης χρησιμοποιώντας παραμέτρους που προσαρμόζονται στις ανάγκες του κάθε ασθενή, υποστηρίζοντας το σωματικό βάρος του και ενισχύοντας την επαναλαμβανόμενη και σταθερή φυσιολογική πρότυπη κίνηση.
- Βελτιστοποιεί τη διαχείριση της θεραπείας, προσφέροντας τις απαραίτητες πληροφορίες και παρακολουθώντας την πρόοδο του ασθενούς.
- Επιτρέπει την προσαρμογή της αντοχής του βάρους χρησιμοποιώντας ρομποτικά πόδια, ανεβάζοντας σταδιακά το φορτίο από τη συσκευή προς τον ασθενή.
- Επιτρέπει στους ασθενείς να συμμετέχουν πιο γρήγορα σε μια ελεγχόμενη μέθοδο επανεκπαίδευσης της βάδισης.

Προηγμένη διαχείριση λογισμικού

Το ρομποτικό σύστημα αποκατάστασης Reo Ambulator είναι εύκολο στη χρήση του, μέσω ενός λογισμικού πολύπλευρης θεραπείας, το οποίο είναι σχεδιασμένο να καταχωρεί τα αποτελέσματα κάθε θεραπείας για κάθε ασθενή.

- Το σύστημα επιτρέπει στους θεραπευτές να διαμορφώνουνε τις ρυθμίσεις αυξάνοντας τα πλεονεκτήματα προς τον ασθενή οδηγώντας τον σε βελτιστοποιημένη εκπαίδευση βάδισης και μειωμένη δυσλειτουργία.

Προηγμένο σύστημα θεραπείας-εκπαίδευσης

- Οι προηγμένες μονάδες θεραπείας-εκπαίδευσης είναι σχεδιασμένες ώστε να προσομοιώνουν, μέσω εικονικής πραγματικότητας, αληθινές καθημερινές καταστάσεις.

Μονάδα Ανάλυσης βάδισης

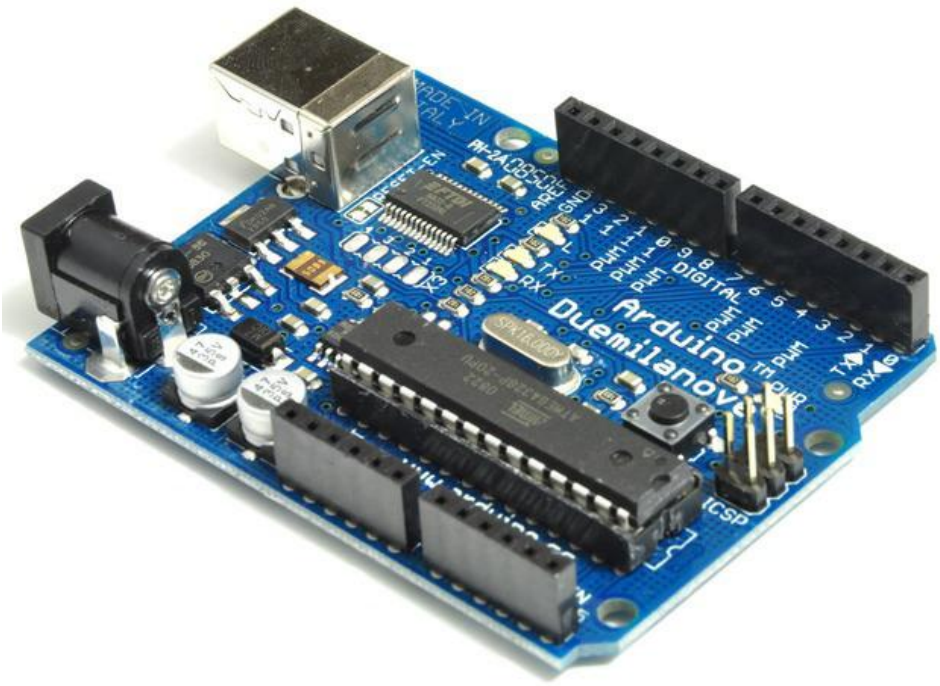
Η καινοτόμος τεχνολογία του Reo Ambulator για τη βάδιση επικεντρώνεται στο να παρακολουθεί το βάδισμα και να δίνει μια ακριβή και σε πραγματικό χρόνο οπτική και ακουστική εικόνα στον ασθενή. Η εικόνα αυτή αποδεικνύεται πως τον βοηθά να βελτιώσει την κίνησή του. Η τεχνολογία αυτή σχεδιάστηκε για ασθενείς που έχουν τη δυνατότητα να περπατούν χωρίς τα ρομποτικά πόδια, με τη χρήση κάποιου άλλου υποστηρικτικού μηχανισμού. Στην οθόνη που βρίσκεται μπροστά στον ασθενή, φαίνονται οι επιθυμητές θέσεις και οι θέσεις που βρίσκονται τα πόδια του ασθενή. Με αυτό τον τρόπο του δίνεται η δυνατότητα να διορθώσει τα μελλοντικά του βήματα. Οι θεραπευτές μπορούν να προσαρμόσουν το μέγεθος και τη συχνότητα ενός βήματος κατά τη διάρκεια μιας άσκησης. Στο τέλος εμφανίζεται μια βαθμολογία που αφορά την ακρίβεια που πέτυχε ο ασθενής. Τα δεδομένα αυτά αποθηκεύονται και είναι διαθέσιμα για τη συνέχιση της παρακολούθησης κάθε ασθενή.

Εικόνα 2.1 Ρομποτικό Σύστημα Υποβοήθησης Κάτω Άκρων REOAMBULATOR

ΚΕΦΑΛΑΙΟ 3. Η ΠΛΑΤΦΟΡΜΑ ARDUINO

3.1 Τι είναι το Arduino

Όπως το περιγράφει ο δημιουργός του, το Arduino είναι μια «ανοικτού κώδικα» πλατφόρμα πρωτοτυποποίησης ηλεκτρονικών, βασισμένη σε ευέλικτο και εύκολο στη χρήση hardware και software, που προορίζεται για οποιονδήποτε έχει λίγη προγραμματιστική εμπειρία, στοιχειώδεις γνώσεις ηλεκτρονικών και ενδιαφέρεται να δημιουργήσει διαδραστικά αντικείμενα ή περιβάλλοντα.

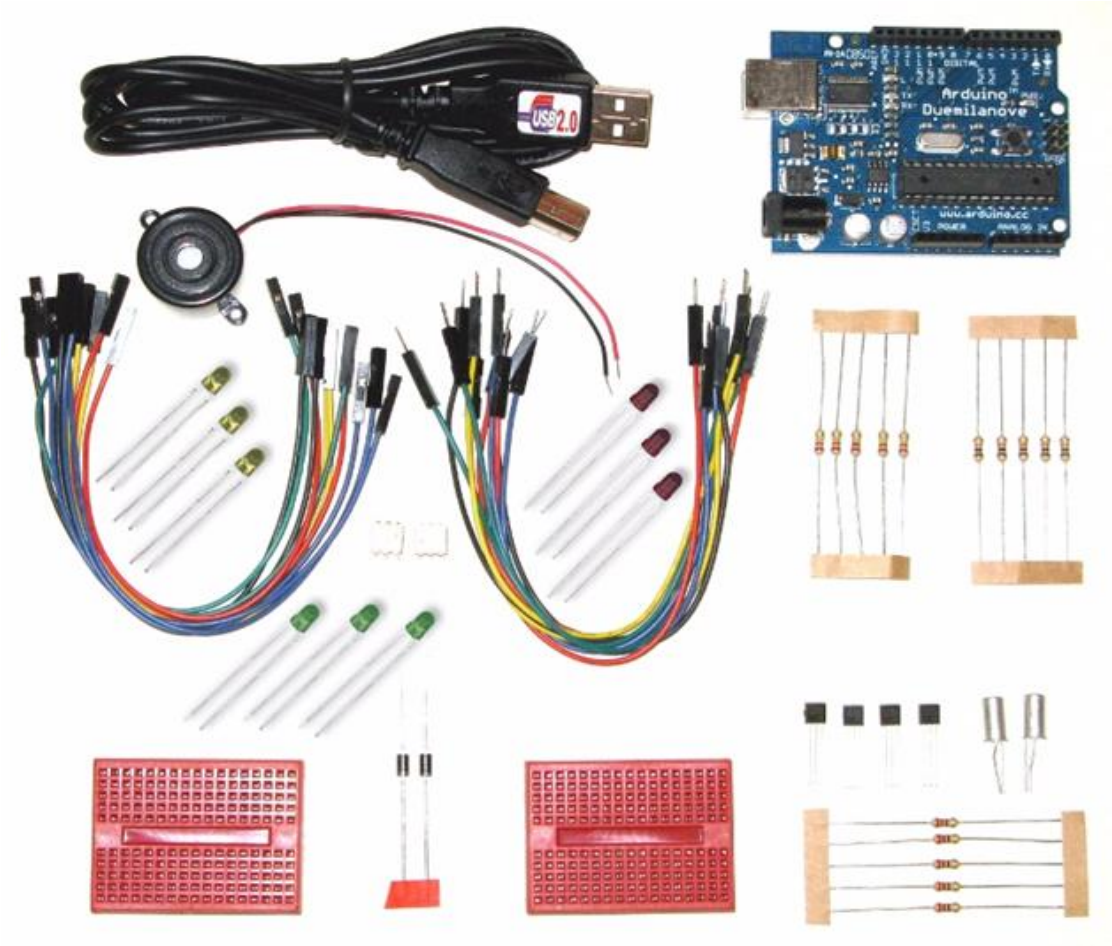


Εικόνα 3.1. Πλατφόρμα Arduino Duemilanova

Στην ουσία, πρόκειται για ένα ηλεκτρονικό κύκλωμα που βασίζεται στον μικροελεγκτή ATmega της Atmel, του οποίου όλα τα σχέδια, καθώς και το software διανέμονται ελεύθερα και δωρεάν, ώστε να μπορεί να κατασκευαστεί από τον καθένα (εξ ού και ο περίεργος -για hardware- χαρακτηρισμός «ανοικτού κώδικα»). Αφού κατασκευαστεί, μπορεί να συμπεριφερθεί σαν ένας μικροσκοπικός υπολογιστής. Ο χρήστης μπορεί

να συνδέσει επάνω του πολλαπλές μονάδες εισόδου-εξόδου, να προγραμματίσει τον μικροελεγκτή να δέχεται δεδομένα από τις μονάδες εισόδου, να τα επεξεργάζεται και να στέλνει κατάλληλες εντολές στις μονάδες εξόδου.

Το Arduino βέβαια, δεν είναι ούτε ο μοναδικός, ούτε ο καλύτερος δυνατός τρόπος για την δημιουργία μιας οποιασδήποτε διαδραστικής ηλεκτρονικής συσκευής. Το κύριο πλεονέκτημά του είναι η τεράστια κοινότητα που το υποστηρίζει και η οποία έχει δημιουργήσει, συντηρεί και επεκτείνει μια online γνωσιακή βάση, ανάλογου μεγέθους.



Εικόνα 3.2. Βασικά Εξαρτήματα Πλατφόρμας Arduino

3.2. Μικροελεγκτής – η καρδιά του Arduino

Το Arduino βασίζεται στον ATmega328, έναν 8-bit RISC μικροελεγκτή, τον οποίο χρονίζει στα 16MHz. Ο ATmega328 διαθέτει ενσωματωμένη μνήμη τριών τύπων:

- **2Kb μνήμης SRAM** είναι η ωφέλιμη μνήμη που μπορούν να χρησιμοποιήσουν οι κώδικες για να αποθηκεύσουν μεταβλητές, πίνακες, κ.λπ. κατά το runtime. Όπως και σε έναν υπολογιστή, αυτή η μνήμη χάνει τα δεδομένα της, όταν η παροχή ρεύματος στο Arduino σταματήσει ή γίνει reset.
- **1Kb μνήμης EEPROM**, η οποία μπορεί να χρησιμοποιηθεί για εγγραφή και ανάγνωση δεδομένων (χωρίς datatype) ανά byte από τον κώδικα κατά το runtime. Σε αντίθεση με την SRAM, η EEPROM δεν χάνει τα περιεχόμενά της με απώλεια τροφοδοσίας ή reset δίσκου.
- **32Kb μνήμης Flash**, από τα οποία τα 2Kb χρησιμοποιούνται από το firmware του Arduino. Το firmware που στην ορολογία του Arduino ονομάζεται bootloader είναι αναγκαίο για την εγκατάσταση του κώδικα μέσω της θύρας USB, χωρίς να χρειάζεται εξωτερικός hardware programmer. Τα υπόλοιπα 30Kb της μνήμης Flash χρησιμοποιούνται για την αποθήκευση του κώδικα, αφού πρώτα μεταγλωττιστεί στον υπολογιστή. Η μνήμη Flash, όπως και η EEPROM δεν χάνει τα περιεχόμενά της, με απώλεια τροφοδοσίας ή reset.

3.3. Διαφορές στις προτεινόμενες εκδόσεις του Arduino

Το πρωτότυπο υλικολογισμικό του Arduino κατασκευάζεται από την Ιταλική εταιρία Smart Projects. Κάποιες πλακέτες με την μάρκα του Arduino έχουν σχεδιαστεί από την Αμερικάνικη εταιρία SparkFun Electronics. Οι διαφορές που έχουν μεταξύ τους οι βασικές πλακέτες Arduino αφορούν συνήθως τον επεξεργαστή και το πλήθος των pins (Input-Output).

Δεκαέξι εκδοχές του Arduino Hardware έχουν χρησιμοποιηθεί εμπορικά μέχρι τώρα οι οποίες παραθέτονται ακολούθως:

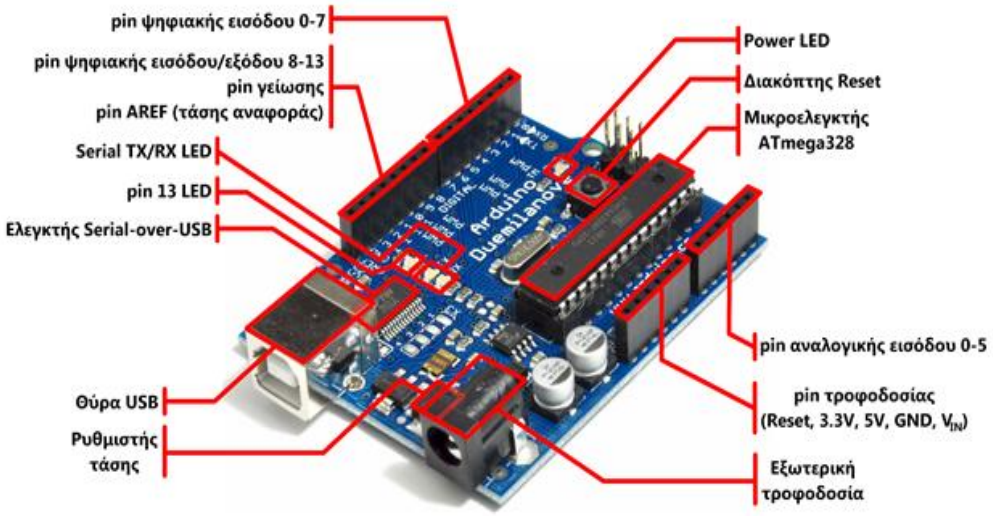
1. Το Serial Arduino, προγραμματισμένο με μία σειριακή DB-9 σύνδεση, χρησιμοποιώντας τεχνολογία ATmega8.
2. Το Arduino Extreme, με USB interface για προγραμματισμό χρησιμοποιώντας τεχνολογία ATmega8.
3. Το Arduino Mini, μία έκδοση μινιατούρας του Arduino χρησιμοποιώντας τεχνολογία surface-mounted ATmega168.
4. Το Arduino Nano, ένα ακόμα πιο μικρό, USB τροφοδοτούμενη εκδοχή του Arduino χρησιμοποιώντας τεχνολογία surface-mounted ATmega168 (ATmega328 για την νεότερη έκδοση).
5. Το LilyPad Arduino, ένα μινιμαλιστικό σχέδιο για εφαρμογές ένδυσης και E-textiles χρησιμοποιώντας τεχνολογία surface-mounted AT-mega328.
6. Το Arduino NG, με ένα USB interface για προγραμματισμό και χρησιμοποιώντας τεχνολογία ATmega8.
7. Το Arduino NG plus, με ένα USB interface για προγραμματισμό και χρησιμοποιώντας τεχνολογία ATmega168.
8. Το Arduino Bluetooth, με Bluetooth interface για προγραμματισμό χρησιμοποιώντας τεχνολογία ATmega168.
9. Το Arduino Diecimila, με ένα USB interface και χρησιμοποιεί τεχνολογία ATmega168 σε ένα DIP28 πακέτο.
10. Το Arduino Duemilanove (“2009”), χρησιμοποιεί τεχνολογία ATmega168 (ATmega328 για την καινούργια έκδοση) και τροφοδοτείται μέσω ενέργειας USB/DC, αυτόματα εναλλασσόμενης.
11. Το Arduino Mega, χρησιμοποιώντας τεχνολογία surface-mounted ATmega1280 για περαιτέρω I/O και μνήμη.
12. Το Arduino Uno, χρησιμοποιώντας την ίδια τεχνολογία ATmega328 όπως το τελευταίο μοντέλο Duemilanove, αλλά ενώ το Duemilanove χρησιμοποιεί ένα FTDI chipset για το USB, το Uno χρησιμοποιεί τεχνολογία ATmega8U2 προγραμματισμένο ως σειριακός μετατροπέας.
13. Το Arduino Mega2560, χρησιμοποιεί τεχνολογία surface-mounted ATmega2560 φέρνοντας την ολική μνήμη στα 256kB. Επίσης ενσωματώνει τη νέα τεχνολογία ATmega8U2, USB chipset.

14. Το Arduino Leonardo, με ένα ATmega32U4 chip που εξαλείφει την ανάγκη για συνδεσιμότητα μέσω USB και μπορεί να χρησιμοποιηθεί ως ψηφιακό πληκτρολόγιο ή ποντίκι.
15. Το Arduino Esplora, με εμφάνιση που παραπέμπει σε χειριστήριο κονσόλας βιντεοπαιχνιδιών με joystick και ενσωματωμένους αισθητήρες για ήχο, φως, θερμοκρασία και επιτάχυνση.
16. Το Arduino Due είναι ένα μικροχειριστήριο board βασισμένο στην τεχνολογία Atmel SAM3X8E ARM Cortex-M3 CPU. Είναι το πρώτο board της Arduino βασισμένη σε επεξεργαστή 32-bit ARM microcontroller.

3.4 Είσοδοι – Έξοδοι

Καταρχήν το Arduino διαθέτει σειριακό interface. Ο μικροελεγκτής ATmega υποστηρίζει σειριακή επικοινωνία, που το Arduino προωθεί μέσα από έναν ελεγκτή Serial-over-USB, ώστε να συνδέεται με τον υπολογιστή μέσω USB.

Η σύνδεση αυτή χρησιμοποιείται για την μεταφορά των προγραμμάτων που σχεδιάζονται από τον υπολογιστή στο Arduino, αλλά και για αμφίδρομη επικοινωνία του Arduino με τον υπολογιστή, μέσα από το πρόγραμμα, την ώρα που εκτελείται.



Εικόνα 3.3. Ακροδέκτες Πλατφόρμας Arduino

Επιπλέον, στην πάνω πλευρά του Arduino βρίσκονται 14 θηλυκά pin, αριθμημένα από 0 ως 13, που μπορούν να λειτουργήσουν ως ψηφιακές είσοδοι και έξοδοι. Λειτουργούν στα 5V και το καθένα μπορεί να παρέχει ή να δεχτεί το πολύ 40mA.

Ως ψηφιακή έξοδος, ένα από αυτά τα pin μπορεί να τεθεί σε κατάσταση HIGH ή LOW από τον κώδικα, οπότε το Arduino θα ξέρει αν πρέπει να διοχετεύσει ή όχι ρεύμα στο συγκεκριμένο pin. Με αυτόν τον τρόπο μπορούμε να ανάψουμε και να σβήσουμε ένα LED που έχουμε συνδέσει στο συγκεκριμένο pin. Αν ορίσω ένα από αυτά τα pin, ως

ψηφιακή είσοδο, μπορούμε με την κατάλληλη εντολή στον κώδικα να διαβάσουμε την κατάστασή του (HIGH ή LOW), ανάλογα με το αν η εξωτερική συσκευή που έχουμε συνδέσει, διοχετεύει ή όχι ρεύμα στο pin. Με αυτόν τον τρόπο μπορούμε να «διαβάσουμε την κατάσταση ενός διακόπτη.

Μερικά από αυτά τα 14 pin, εκτός από ψηφιακές είσοδοι/έξοδοι έχουν και δεύτερη λειτουργία. Συγκεκριμένα:

- Τα pin «0» και «1» λειτουργούν ως RX και TX της σειριακής όταν ο κώδικας ενεργοποιεί την σειριακή θύρα. Έτσι, όταν ο κώδικας στέλνει δεδομένα στην σειριακή, αυτά προωθούνται και στην θύρα USB, μέσω του ελεγκτή Serial-Over-USB αλλά και στο pin «0» για να τα διαβάσει ενδεχομένως μια άλλη συσκευή (π.χ. ένα δεύτερο Arduino στο δικό του pin «1»). Αυτό σημαίνει ότι αν στον κώδικα ενεργοποιήσουμε το σειριακό interface, χάνουμε 2 ψηφιακές εισόδους/εξόδους.
- Τα pin «2» και «3» λειτουργούν και ως εξωτερικά interrupt (interrupt «0» και «1» αντίστοιχα). Μπορούμε να τα ρυθμίσουμε μέσα από τον κώδικα, ώστε να λειτουργούν αποκλειστικά ως ψηφιακές είσοδοι στις οποίες όταν συμβαίνουν συγκεκριμένες αλλαγές να σταματάει άμεσα, η κανονική ροή του προγράμματος και να εκτελείται μια συγκεκριμένη συνάρτηση. Τα εξωτερικά interrupt είναι ιδιαίτερα χρήσιμα σε εφαρμογές που απαιτούν συγχρονισμό μεγάλης ακρίβειας.
- Τα pin «3», «5», «6», «9», «10» και «11» μπορούν να λειτουργήσουν και ως ψευδοαναλογικές έξοδοι με το σύστημα PWM (Pulse Width Modulation), δηλαδή το ίδιο σύστημα που διαθέτουν οι μητρικές των υπολογιστών, για να ελέγχουν τις ταχύτητες των ανεμιστήρων. Έτσι, μπορούμε να συνδέσουμε ένα LED σε κάποιο από αυτά τα pin για να ελέγξουμε πλήρως την φωτεινότητά του, με ανάλυση 8bit (256 καταστάσεις από «0»-σβηστό ως «255»-πλήρως αναμμένο), αντί να έχουμε απλά την δυνατότητα αναμμένο-σβηστό, που παρέχουν οι υπόλοιπες ψηφιακές έξοδοι. Επισημαίνεται ότι το PWM δεν είναι πραγματικά αναλογικό σύστημα και θέτοντας στην έξοδο την τιμή «27», δεν σημαίνει ότι η έξοδος θα δίνει 2.5V αντί της

κανονικής τιμής των 5V, αλλά ότι θα δίνει ένα παλμό που θα εναλλάσσεται με μεγάλη συχνότητα και για ίσους χρόνους μεταξύ των τιμών 0 και 5V.

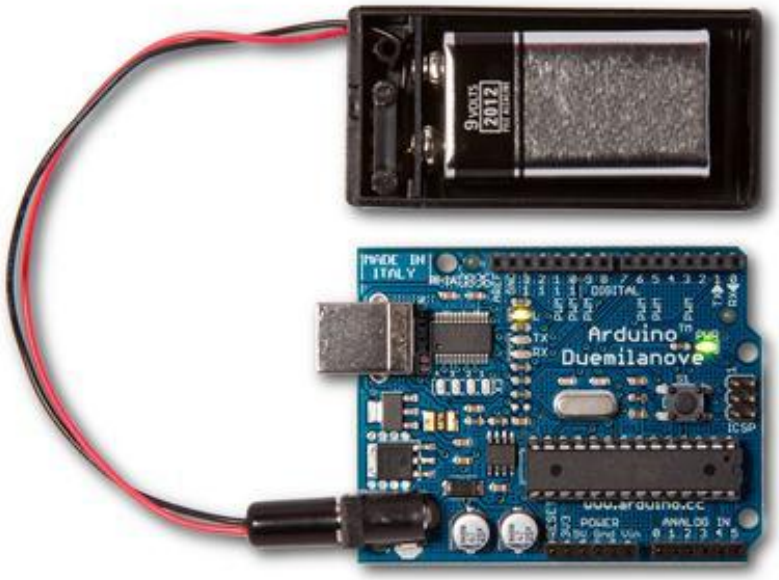
Στην κάτω πλευρά του Arduino, με τη σήμανση ANALOG IN, υπάρχει μια ακόμη σειρά από 6 pin, αριθμημένα από το «0» ως το «5». Το καθένα από αυτά λειτουργεί ως αναλογική είσοδος κάνοντας χρήση του ADC (Analog to Digital Converter), που είναι ενσωματωμένο στον μικροελεγκτή. Για παράδειγμα, μπορούμε να τροφοδοτήσουμε ένα από αυτά με μια τάση, την οποία μπορούμε να κυμάνουμε με ένα ποτενσιόμετρο από 0V (ως μια τάση αναφοράς Vref), η οποία είναι προρυθμισμένη στα 5V. Τότε, μέσα από τον κώδικα μπορούμε να «διαβάσουμε» την τιμή του pin ως ένα ακέραιο αριθμό ανάλυσης 10bit, από «0» (όταν η τάση στο pin είναι 0V) μέχρι «1023» (όταν η τάση στο pin είναι 5V). Η τάση αναφοράς μπορεί να ρυθμιστεί με μια εντολή στο 1.1V, ή σε όποια τάση επιθυμούμε (μεταξύ 2 και 5V), τροφοδοτώντας εξωτερικά με αυτή την τάση το pin με την σήμανση AREF, που βρίσκεται στην απέναντι πλευρά της πλακέτας. Έτσι, αν τροφοδοτήσουμε το pin AREF με 3.3V και στην συνέχεια δοκιμάσουμε να διαβάσουμε κάποιο pin αναλογικής εισόδου, στο οποίο εφαρμόζεται τάση 1.65V, το Arduino θα επιστρέψει την τιμή «512».

Τέλος, καθένα από τα 6 pin, με κατάλληλη εντολή μέσα από τον κώδικα μπορεί να μετατραπεί σε ψηφιακό pin εισόδου/εξόδου, όπως τα 14 pin, που βρίσκονται στην απέναντι πλευρά και τα οποία περιεγράφηκαν πριν. Σε αυτή την περίπτωση τα pin μετονομάζονται από 0~5 σε 14~19 αντίστοιχα.

3.5 Τροφοδοσία

Το Arduino μπορεί να τροφοδοτηθεί με ρεύμα, είτε από τον υπολογιστή, μέσω της σύνδεσης USB, είτε από εξωτερική τροφοδοσία που παρέχεται μέσω μιας υποδοχής φις των 2.1mm (θετικός πόλος στο κέντρο) και βρίσκεται στην κάτω-αριστερή γωνία του Arduino.

Εικόνα 3.4. Τροφοδοσία Πλατφόρμας Arduino μέσω τροφοδοτικού 220V AC



Εικόνα 3.5. Τροφοδοσία Πλατφόρμας Arduino μέσω μπαταρίας 9V

Η εξωτερική τροφοδοσία του Arduino κυμαίνεται από 7 ως 12V και μπορεί να προέρχεται από ένα κοινό μετασχηματιστή του εμπορίου, από μπαταρίες ή οποιαδήποτε άλλη πηγή DC.

Υπάρχει μια ακόμα συστοιχία από 6 pin με την σήμανση POWER, δίπλα από τα pin της αναλογικής εισόδου. Η λειτουργία του καθενός έχει ως εξής:

- Το πρώτο, με την ένδειξη RESET, όταν γειωθεί (σε οποιοδήποτε από τα 3 pin με την ένδειξη GND, που υπάρχουν στο Arduino) έχει ως αποτέλεσμα την επανεκκίνηση του Arduino.
- Το δεύτερο, με την ένδειξη 3.3V τροφοδοτεί τα εξαρτήματα με τάση 3.3V. Η τάση αυτή δεν προέρχεται από την εξωτερική τροφοδοσία, αλλά παράγεται από τον ελεγκτή Serial-over-USB και έτσι η μέγιστη ένταση που μπορεί να παρέχει είναι μόλις 50mA.
- Το τρίτο, με την ένδειξη 5V, τροφοδοτεί τα εξαρτήματα με τάση 5V. Ανάλογα με τον τρόπο τροφοδοσίας του ίδιου του Arduino, η τάση αυτή προέρχεται είτε άμεσα από την θύρα USB (που ούτως ή άλλως λειτουργεί στα 5V), είτε από την εξωτερική τροφοδοσία, αφού αυτή περάσει από ένα ρυθμιστή τάσης για να την «φέρει» στα 5V.
- Το τέταρτο και το πέμπτο pin, με την ένδειξη GND, είναι γειώσεις.
- Το έκτο pin, με την ένδειξη Vin έχει διπλό ρόλο:
  α. λειτουργεί ως μέθοδος εξωτερικής τροφοδοσίας του Arduino, σε συνδυασμό με το pin γείωσης δίπλα του, στην περίπτωση που δεν χρησιμοποιείται η υποδοχή του φις των 2.1mm.
  β. όταν υπάρχει συνδεδεμένη εξωτερική τροφοδοσία μέσω του φις, το pin χρησιμοποιείται για να τροφοδοτούνται εξαρτήματα με την πλήρη τάση της εξωτερικής τροφοδοσίας (7~12V), πριν αυτή περάσει από τον ρυθμιστή τάσης, όπως γίνεται με το pin των 5V.

3.6. Ενσωματωμένα κουμπιά και LED

Πάνω στην πλακέτα του Arduino υπάρχει ένας διακόπτης micro-switch και 4 μικροσκοπικά LED επιφανειακής στήριξης.

Η λειτουργία του διακόπτη RESET έχει ως σκοπό την επανεκκίνηση του επεξεργαστή.

Το LED με την ένδειξη «ON» χρησιμοποιείται ως ένδειξη σωστής τροφοδοσίας.

Τα δύο LED με τις σημάνσεις TX και RX, χρησιμοποιούνται ως ένδειξη λειτουργίας του σειριακού interface, καθώς ανάβουν όταν το Arduino στέλνει ή λαμβάνει δεδομένα μέσω USB. Τα LED αυτά ελέγχονται από τον ελεγκτή Serial-over-USB και δεν λειτουργούν, όταν η σειριακή επικοινωνία γίνεται αποκλειστικά μέσω των ψηφιακών pin 0 και 1.

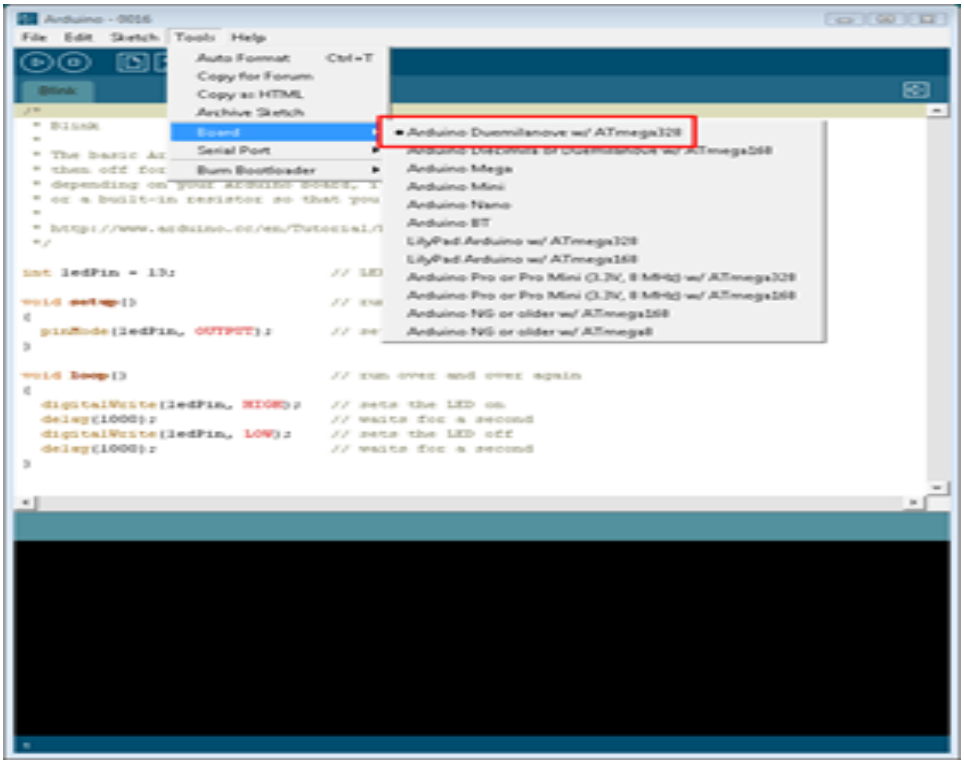
Η βασική δοκιμή λειτουργίας του Arduino είναι να του αναθέσουμε να αναβοσβήνει ένα LED. Για να μπορούμε να το κάνουμε αυτό, χωρίς να συνδέσουμε τίποτα πάνω στο Arduino, οι κατασκευαστές του σκέφτηκαν να ενσωματώσουν ένα LED(Με σήμανση L), το οποίο σύνδεσαν στο ψηφιακό pin 13. Έτσι, ακόμα και αν δεν έχουμε συνδέσει τίποτα πάνω στο φυσικό pin 13, αναθέτοντάς του την τιμή HIGH μέσα από τον κώδικα, θα ανάψει αυτό το ενσωματωμένο LED.

3.7. Arduino IDE και σύνδεση με τον υπολογιστή

Τα απαραίτητα για την διαχείριση του Arduino από τον υπολογιστή παρέχονται από το Arduino IDE.

Το Arduino IDE είναι βασισμένο σε Java και παρέχει συγκεκριμένα:

- ένα πρακτικό περιβάλλον για την συγγραφή του κώδικα (τα οποία ονομάζονται sketch στην ορολογία του Arduino) με συντακτική χρωματική σήμανση,
- αρκετά έτοιμα παραδείγματα,
- μερικές έτοιμες βιβλιοθήκες για προέκταση της γλώσσας και τον ευκολότερο χειρισμό των εξαρτημάτων που συνδέονται στο Arduino,
- τον compiler για την μεταγλώττιση των sketch σας,
- ένα serial monitor που παρακολουθεί τις επικοινωνίες της σειριακής (USB), αναλαμβάνει να στείλει αλφαριθμητικά δεδομένα στο Arduino, μέσω αυτής και είναι ιδιαίτερα χρήσιμο για το debugging των sketch.
- την επιλογή να ανεβάζουμε το μεταγλωττισμένο sketch στο Arduino.



Εικόνα 3.6. Περιβάλλον Ανάπτυξης Λογισμικού IDE

3.8. Γλώσσα προγραμματισμού

Η γλώσσα του Arduino βασίζεται στη γλώσσα Wiring, μια παραλλαγή C/C++ για μικροελεγκτές αρχιτεκτονικής AVR όπως ο ATmega, και υποστηρίζει όλες τις βασικές δομές της C καθώς και μερικά χαρακτηριστικά της C++. Για compiler χρησιμοποιείται ο AVR gcc και ως βασική βιβλιοθήκη C χρησιμοποιείται η AVR libc.

Λόγω της καταγωγής της από την C, στην γλώσσα του Arduino μπορούμε να χρησιμοποιήσουμε ουσιαστικά τις ίδιες βασικές εντολές και συναρτήσεις, με την ίδια σύνταξη, τους ίδιους τύπων δεδομένων και τους ίδιους τελεστές όπως και στην C. Πέρα από αυτές όμως, υπάρχουν κάποιες ειδικές εντολές, συναρτήσεις και σταθερές που βοηθούν για την διαχείριση του ειδικού hardware του Arduino. Οι πιο σημαντικές από αυτές αναλύονται στον πίνακα που ακολουθεί:

Όρισμα	Είδος	Τύπος	Παράμετροι	Περιγραφή
LOW	Σταθερά	int	–	Έχει την τιμή 0 και είναι αντίστοιχη του λογικού false.
HIGH	Σταθερά	int	–	Έχει την τιμή 1 και είναι αντίστοιχη του λογικού true.
INPUT	Σταθερά	int	–	Έχει την τιμή 0 και είναι αντίστοιχη του λογικού false.
OUTPUT	Σταθερά	int	–	Έχει την τιμή 1 και είναι αντίστοιχη του λογικού true.
pinMode	Εντολή	–	(*pin*, *mode*)	Καθορίζει αν το συγκεκριμένο ψηφιακό *pin* θα είναι pin εισόδου ή pin εξόδου ανάλογα με την τιμή που δίνεται στην παράμετρο *mode*(INPUT ή OUTPUT αντίστοιχα).
digitalWrite	Εντολή	–	(*pin*,*pinstatus*)	Θέτει την κατάσταση *pinstatus*(HIGH ή LOW) στο συγκεκριμένο ψηφιακό *pin*.
digitalRead	Συνάρτηση	int	(*pin*)	Επιστρέφει την κατάσταση του συγκεκριμένου ψηφιακού *pin* (0 για LOW και 1 για HIGH) εφόσον αυτό είναι pin εισόδου.

Όρισμα	Είδος	Τύπος	Παράμετροι	Περιγραφή
analogWrite	Εντολή	–	(*pin*, *value*)	Θέτει το συγκεκριμένο ψηφιακό *pin* σε κατάσταση ψευδοαναλογικής εξόδου (PWM). Η παράμετρος *value* καθορίζει το πλάτος του παλμού σε σχέση με την περίοδο του παραγόμενου σήματος στην κλίμακα από 0 ως 255 (π.χ. με *value* 127, το πλάτος του παλμού είναι ίσο με μισή περίοδο).
millis	Συνάρτηση	unsigned long	()	Μετρητής που επιστρέφει το χρονικό διάστημα σε ms από την στιγμή που άρχισε η εκτέλεση του προγράμματος. Λάβετε υπόψη ότι λόγω του τύπου μεταβλητής (unsigned long δηλ. 32bit) θα γίνει overflow σε 2^32ms δηλαδή περίπου σε 50 μέρες, οπότε ο μετρητής θα ξεκινήσει πάλι από το μηδέν.
delay	Εντολή	–	(*time*)	Σταματά προσωρινά την ροή του προγράμματος για *time* ms. Η παράμετρος *time* είναι unsigned long (από 0 ως 2^32). Σημειώστε ότι παρά την προσωρινή παύση, συναρτήσεις των οποίων η εκτέλεση ενεργοποιείται από interrupt θα εκτελεστούν κανονικά κατά την διάρκεια μιας delay.
attachInterrupt	Εντολή	–	(*interrupt*, *function*, *triggermode*)	Θέτει σε λειτουργία το συγκεκριμένο interrupt, ώστε να ενεργοποιεί την συνάρτηση(function), κάθε φορά που ικανοποιείται η συνθήκη που ορίζεται από την παράμετρο triggermode: LOW :ενεργοποίηση όταν η κατάσταση του pin που αντιστοιχεί στο συγκεκριμένο interrupt γίνει LOW. RISING :όταν από LOW γίνει HIGH, FALLING: όταν από HIGH γίνει LOW.

				CHANGE: όταν αλλάξει η κατάσταση γενικά.

Όρισμα	Είδος	Τύπος	Παράμετροι	Περιγραφή
detachInterrupt	Εντολή	–	(*interrupt*)	Απενεργοποιεί το συγκεκριμένο *interrupt*.
noInterrupts	Εντολή	–	()	Σταματά προσωρινά την λειτουργία όλων των interrupt
interrupts	Εντολή	–	()	Επαναφέρει την λειτουργία των interrupt που διακόπηκε προσωρινά από μια εντολή noInterrupts.
Serial.begin	Μέθοδος κλάσης	–	(*datarate*)	Θέτει τον ρυθμό μεταφοράς δεδομένων του σειριακού interface (σε baud)
Serial.println	Μέθοδος κλάσης	–	(*data*)	Διοχετεύει τα δεδομένα *data* για αποστολή μέσω του σειριακού interface. Η παράμετρος *data* μπορεί να είναι είτε αριθμός είτε αλφαριθμητικό.

Επιπλέον, στην γλώσσα του Arduino κάθε πρόγραμμα αποτελείται από δύο βασικές ρουτίνες ώστε να έχει την γενική δομή:

```
// Ενσωματώσεις βιβλιοθηκών, δηλώσεις μεταβλητών...
void setup()
{
  // ...
}
void loop()
{
  // ...
}
// Υπόλοιπες συναρτήσεις...
```

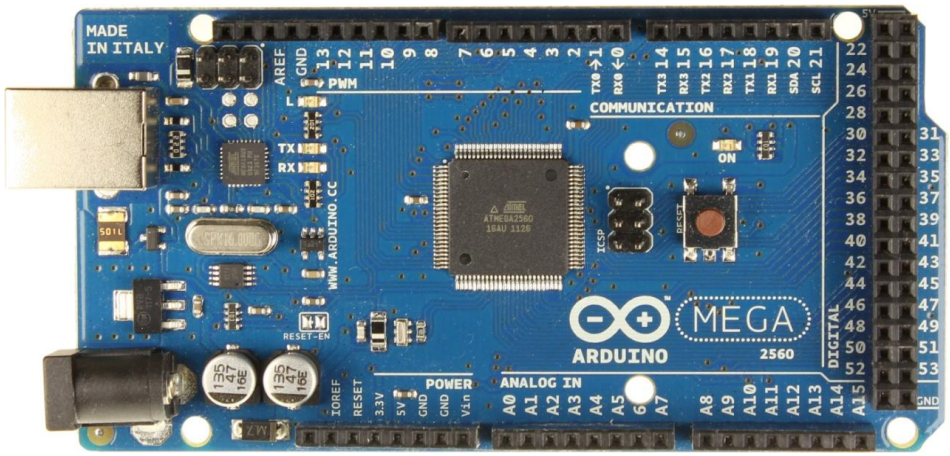
Η βασική ρουτίνα setup() εκτελείται μια φορά μόνο κατά την εκκίνηση του προγράμματος, ενώ η βασική ρουτίνα loop() περιέχει τον βασικό κορμό του προγράμματος και η εκτέλεσή της επαναλαμβάνεται συνέχεια σαν ένας βρόγχος while(true).

ΚΕΦΑΛΑΙΟ 4. ΗΛΕΚΤΡΟΝΙΚΑ ΜΕΡΗ

4.1. Ηλεκτρονικά εξαρτήματα

4.1.1. Arduino Mega 2560

Το Arduino Mega 2560 είναι η υπολογιστική πλατφόρμα που αποτελείται από τον μικροελεγκτή ATMEGA 2560, όπως αναφέρεται στο προηγούμενο κεφάλαιο.



Εικόνα 4.1. Πλατφόρμα Arduino Mega 2560

4.1.2. Graphic LCD Screen ST 7920

Η οθόνη είναι διαστάσεων 70 χ 38 χιλ., χρώματος μπλέ με λευκά γράμματα και αποτελείται από 128 χ 64 pixels.

Για την σύνδεσή της χρησιμοποιήθηκε το σειριακό πρωτόκολλο SPI, το οποίο απαιτεί τρεις αγωγούς.

Η λειτουργία της είναι πολύ σημαντική, διότι επιτρέπει την διεπαφή του ασθενούς με το όργανο εκγύμνασης. Ο ασθενής μπορεί εύκολα αφενός να πληροφορηθεί το score του και αφετέρου να παρακολουθεί τις επιτευχθείσες μοίρες σε κάθε άρθρωση, κάθε φορά που ασκείται. Οι πληροφορίες αυτές λειτουργούν καταλυτικά στην ψυχολογία και στην εξέλιξη της απόδοσης του, διότι μεταβάλλονται σε κίνητρο παρότρυνσης ταχείας ανάρρωσης.

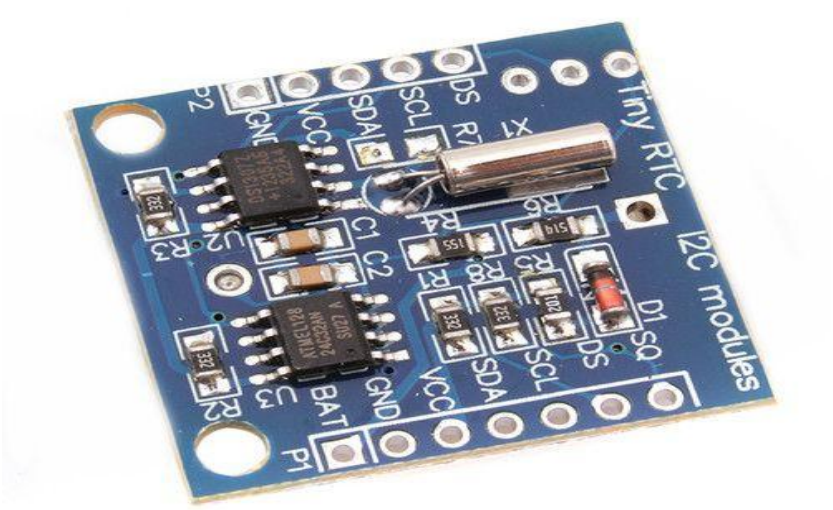
Εικόνα 4.2. Οθόνη γραφικών ST 7920

4.1.3. Real Time Clock DS 1307

Είναι ένα σειριακό ρολόϊ πραγματικού χρόνου της MAXIM σε διάδικό κώδικα και παρέχει πληροφορίες δευτερολέπτων, λεπτών, ώρας και ημερομηνίας. Το ρολόϊ επικοινωνεί με την υπολογιστική πλατφόρμα Arduino, μέσω του πρωτοκόλλου i2c για την μεταφορά δεδομένων (η αγωγή SCL και SDA).

Το real time clock διαθέτει μία εφεδρική μπαταρία (CR1220) προκειμένου να διατηρεί την ώρα, όταν δεν υπάρχει τροφοδοσία στο κύκλωμα.

Το σειριακό ρολόϊ είναι απολύτως απαραίτητο για την καταγραφή της ημερομηνίας και ώρας της άσκησης εκάστου ασθενούς, ώστε να παρακολουθείται η εξέλιξη της ανάρρωσής του.

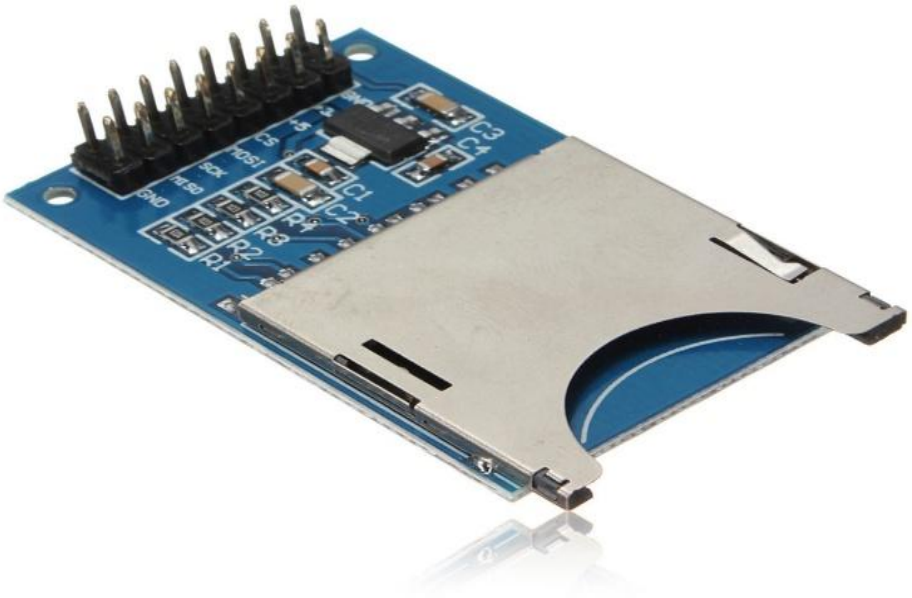


Εικόνα 4.3. Ρολόϊ πραγματικού χρόνου DS 1307

4.1.4. Κάρτα Αποθήκευσης SD

Η υποδοχή είναι συνδεδεμένη στην πλατφόρμα Arduino, μέσω του buffer IC3. Ο buffer είναι ένας level shifter και μετατρέπει τα σήματα των 5V σε σήματα των 3,3V, τα οποία είναι πιο ασφαλή για την κάρτα SD σε σχέση με αυτά των 5V.

Η κάρτα SD χρησιμοποιείται για την αποθήκευση αρχείων excel, όπου αποθηκεύονται τα ιστορικά των ασθενών, ώστε να είναι εφικτή η έκδοση στατιστικών μετρήσεων.



Εικόνα 4.4. Κράτα αποθήκευσης SD

4.1.5. Πιεζοηλεκτρικός Βομβητής

Ο πιεζοηλεκτρικός βομβητής δέχεται τάση 5V και ελέγχεται μέσω ενός digital pin της πλατφόρμας του Arduino. Ο ρόλος του πιεζοηλεκτρικού βομβητή είναι σημαντικός, διότι παροτρύνει τον ασθενή να μην εγκαταλείπει την προσπάθεια και τον επιβραβεύει σε κάθε προσπάθεια άσκησης αυτού.

Εικόνα 4.5. Πιεζοηλεκτρικός βομβητής

4.1.6. Πληκτρολόγιο 4 x 4

Το πληκτρολόγιο αποτελείται από 16 πλήκτρα, εκ των οποίων τα 10 είναι αριθμοί , 4 γράμματα (A,B,C,D) και δύο σύμβολα (*, #). Το πληκτρολόγιο επιτρέπει στον χρήστη να περιηγείται στο menu, αλλά και να εισάγει το user name, προκειμένου να αρχίσει την άσκηση.

Εικόνα 4.6. Πληκτρολόγιο 4 x 4

4.1.7. Ποτενσιόμετρο

Το ποτενσιόμετρο είναι αναλογικό ηλεκτρονικό εξάρτημα που χρησιμοποιείται στα κυκλώματα ως μεταβλητή αντίσταση. Αποτελείται από αγώγιμη πλάκα σχήματος «Ω», πάνω στην οποία γυρίζει μία επαφή με την βοήθεια ενός στροφέα. Ανάλογα με την απόσταση της επαφής από την είσοδο του ρεύματος στο ποτενσιόμετρο μεταβάλλεται και η αντίσταση.

Στην πραγματικότητα είναι ένας καταμεριστής τάσης μόνο που έχει μεταβλητή σχέση καταμερισμού και χρησιμεύει σε μετρήσεις χαμηλών τάσεων. Πρακτικά αποτελείται από μία μεταβλητή αντίσταση, πάνω στην οποία κινείται ένας δρομέας, η κίνηση του οποίου ρυθμίζεται από τον χρήστη.

Το ποτενσιόμετρο είναι καθοριστικό στην κατασκευή του οργάνου εκγύμνασης, διότι με αυτό μπορούμε να μετρήσουμε τις επιτευχθείσες μοίρες σε κάθε άρθρωση.

Εικόνα 4.7. Ποτενσιόμετρο

4.1.8. Τροφοδοτικό

Το τροφοδοτικό είναι μία εξωτερική ηλεκτρονική συσκευή που μετατρέπει την τάση του δικτύου (220 ή 110 Volt) στην απαιτούμενη τάση και τύπο ρεύματος (εναλλασσόμενο ή συνεχές), που είναι κατάλληλο για τη λειτουργία μιας ηλεκτρονικής συσκευής.

Στην συγκεκριμένη περίπτωση το όργανο εκγύμνασης δέχεται τάση 7 έως 12 Volt συνεχούς ρεύματος, συνεπώς χρησιμοποιήθηκε τροφοδοτικό συνεχούς ρεύματος 12Volt 2A.

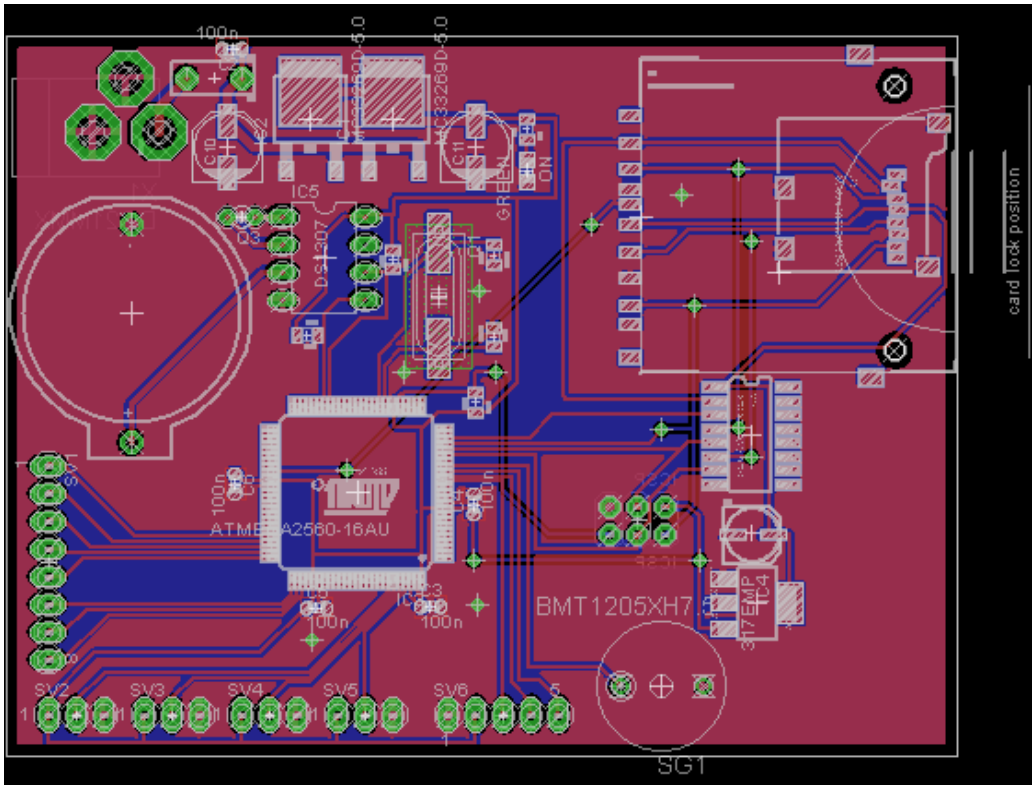
Εικόνα 4.8. Τροφοδοτικό συνεχούς ρεύματος 12Volt 2A

4.2. Ηλεκτρονικό κύκλωμα

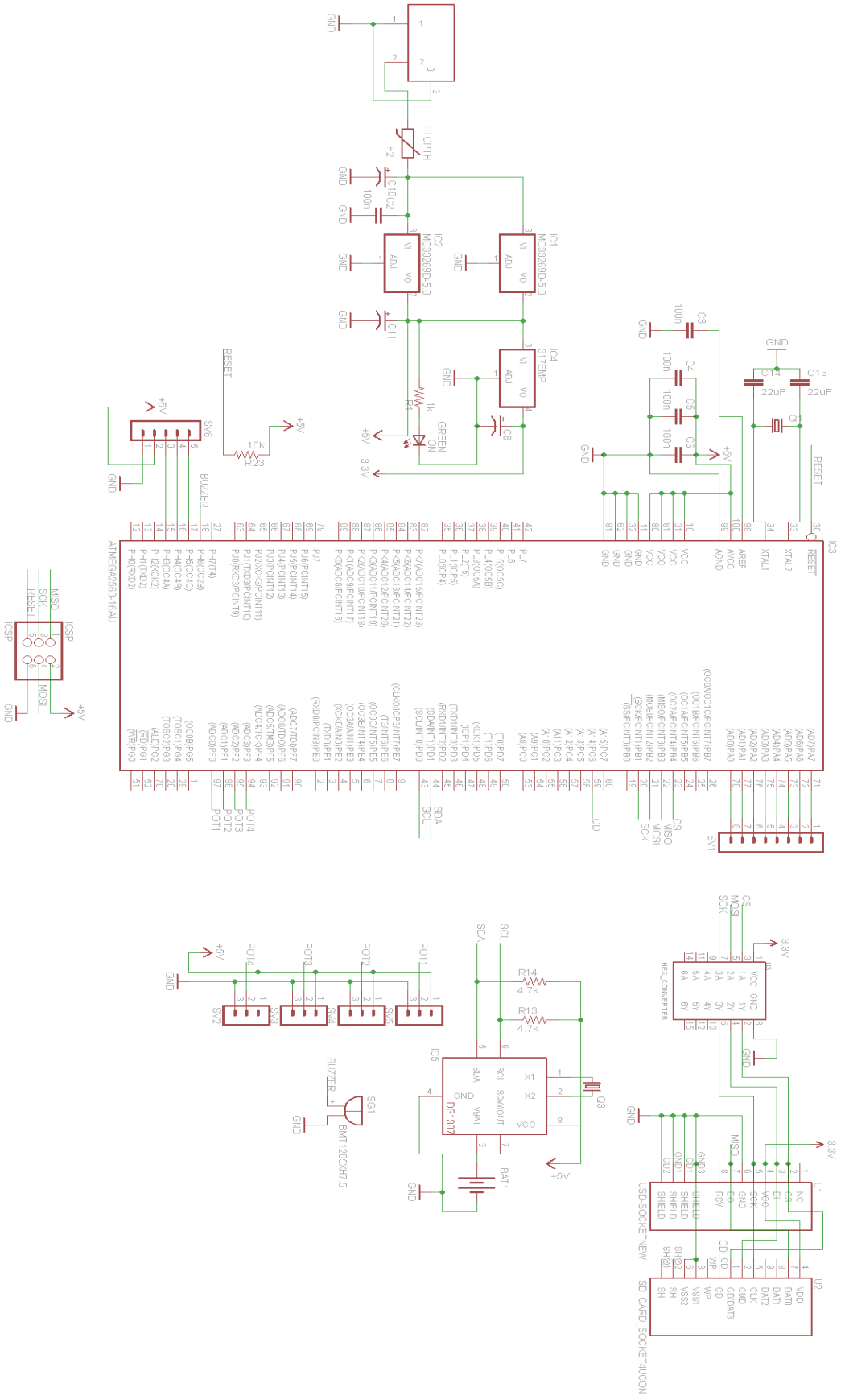
Το ηλεκτρονικό κύκλωμα σχεδιάστηκε χρησιμοποιώντας το πρόγραμμα Eagle της Cadsoft.

Στο κύκλωμα αυτό προστέθηκε ο επεξεργαστής ATMEGA 2560 της ATMEL με ένα κρύσταλλο 16MHz και έγιναν οι απαραίτητες συνδέσεις για την οδήγησή του. Επιπλέον, το κύκλωμα διαθέτει :

- ένα voltage regulator το οποίο δέχεται τάση από 7-12V και τα μετατρέπει σε 5V σταθερά,
- ολοκληρωμένο κύκλωμα ανάγνωσης ώρας (ds1307),
- ολοκληρωμένο κύκλωμα ανάγνωσης κάρτας αποθήκευσης,
- ακροδέκτες ποτενσιομέτρων,
- ακροδέκτες οθόνης.



Εικόνα 4.9. Πλακέτα οργάνου εκγύμνασης



Εικόνα 4.10. Σχηματικό πλακέτας οργάνου εκγύμνασης

ΚΕΦΑΛΑΙΟ 5. ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΗΣ

5.1. Λειτουργίες μικροϋπολογιστή

Ο μικροϋπολογιστής διαχωρίζει την πλατφόρμα από ένα απλό όργανο εκγύμνασης. Έχει δε τοποθετηθεί στη δεξιά πλευρά της κατασκευής (ύψος 65 cm), ώστε να επιτρέπει στον χρήστη να επικοινωνεί μαζί του, παρακολουθώντας το score κάθε συνεδρίας.

Αυτά τα δεδομένα είναι εξαιρετικά σημαντικά και ωφέλιμα για τον ασθενή, διότι:

α. παρακολουθεί την εξέλιξη της ανάρρωσής του, εκ του σύνεγγυς,

β. παροτρύνεται να συνεχίσει την προσπάθεια ανάκαμψης,

γ. είναι εφικτή η έκδοση στατιστικών δεδομένων μίας εκάστης συνεδρίας.

Βασικές λειτουργίες του μικροϋπολογιστή αποτελεί :

- Η μέτρηση των επιτευχθέντων μοιρών σε κάθε άρθρωση.
- Ο υπολογισμός του score από το άθροισμα της διαφοράς των μέγιστων και ελάχιστων της κάθε άρθρωσης (Score = ($Max_{άρθρωσης1}$ – $Min_{άρθρωσης1}$) + ($Max_{άρθρωσης2}$ – $Min_{άρθρωσης2}$) + ($Max_{άρθρωσης3}$ – $Min_{άρθρωσης3}$) + ($Max_{άρθρωσης4}$ – $Min_{άρθρωσης4}$)).
- Η επιβράβευση του χρήστη μέσω ηχητικών τόνων, όταν επιτευχθεί νέο score.
- Η καταγραφή δεδομένων και ημερομηνίας της συνεδρίας κάθε ασθενούς. Η αποθήκευση των δεδομένων κάθε ασθενή γίνεται σε μοναδικό φύλλο excel, προκειμένου να εξεταστούν περαιτέρω τα αποτελέσματα από τον γιατρό.

5.2. Περιγραφή Μενού

Ο μικροϋπολογιστής περιέχει πέντε (5) διαφορετικά μενού που περιλαμβάνουν πληροφορίες για τον χρήστη και ρυθμίσεις της πλατφόρμας.

Κεντρικό Μενού

Στο Μενού αυτό ο ασθενής μπορεί να ενημερωθεί για τις επιτευχθείσες μοίρες σε κάθε άρθρωση, αλλά και το score που έχει επιτύχει. Πιο συγκεκριμένα στην πάνω δεξιά γωνία της οθόνης αναγράφονται οι μοίρες του δεξιού αγκώνα. Αντίστοιχα στην πάνω αριστερή γωνία εμφανίζονται οι μοίρες του αριστερού αγκώνα. Στην κάτω δεξιά γωνία εμφανίζονται οι μοίρες του δεξιού ώμου και στην αριστερή γωνία του αριστερού ώμου, αντίστοιχα.

Στο κέντρο της οθόνης εμφανίζεται το score του ασθενή. Πάνω από το score εμφανίζεται ο κωδικός του ασθενή, αν έχει οριστεί, άλλως εμφανίζεται το μήνυμα «no user». Κάτω από το score εμφανίζεται η ώρα και η ημερομηνία της άσκησης.

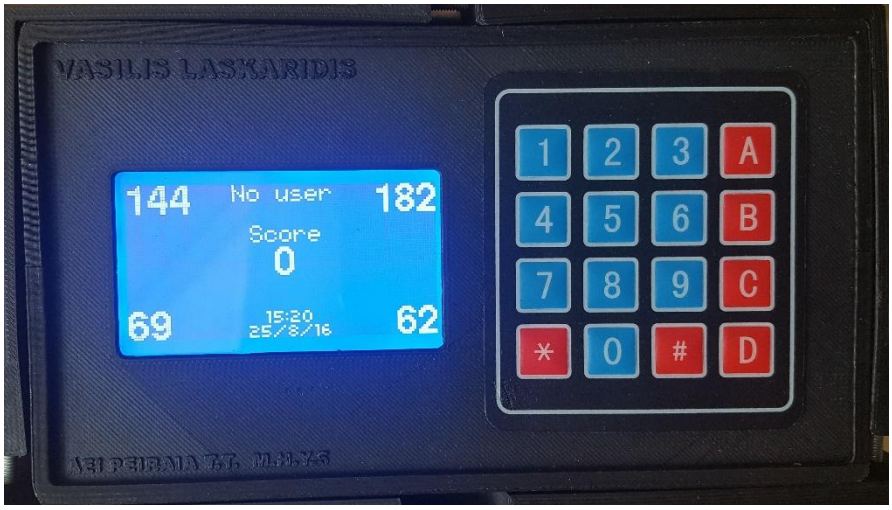
Τα πλήκτρα A, B, C, D επιτρέπουν στον χρήστη να περιηγηθεί στα υπόλοιπα μενού του μικροϋπολογιστή που αφορούν :

A: Εισαγωγή κωδικού ασθενή,

B: Διακοπή άσκησης του συγκεκριμένου ασθενή,

C: Ρυθμίσεις μικροϋπολογιστή και

D: Προβολή δεδομένων ασθενούς



Εικόνα 5.1. Κεντρικό Μενού μικροϋπολογιστή

Μενού Εισαγωγής κωδικού ασθενή

Ο κωδικός του ασθενή εισάγεται στο μενού, προκειμένου να είναι εφικτή η καταγραφή των δεδομένων του ιστορικού του. Με τους αριθμούς που βρίσκονται στα πλήκτρα εισάγεται ο κωδικός του ασθενή (5 ψηφία) και πιέζοντας το πλήκτρο «A», αποθηκεύεται και επιστρέφει στο κεντρικό μενού. Σε περίπτωση που δεν είναι επιθυμητή η αποθήκευση των στοιχείων του ασθενούς πιέζουμε το πλήκτρο «B» και επιστρέφουμε στο κεντρικού μενού, χωρίς αποθήκευση του κωδικού. Σε περίπτωση λάθους κωδικού, ακυρώνουμε πιέζοντας το πλήκτρο «C».

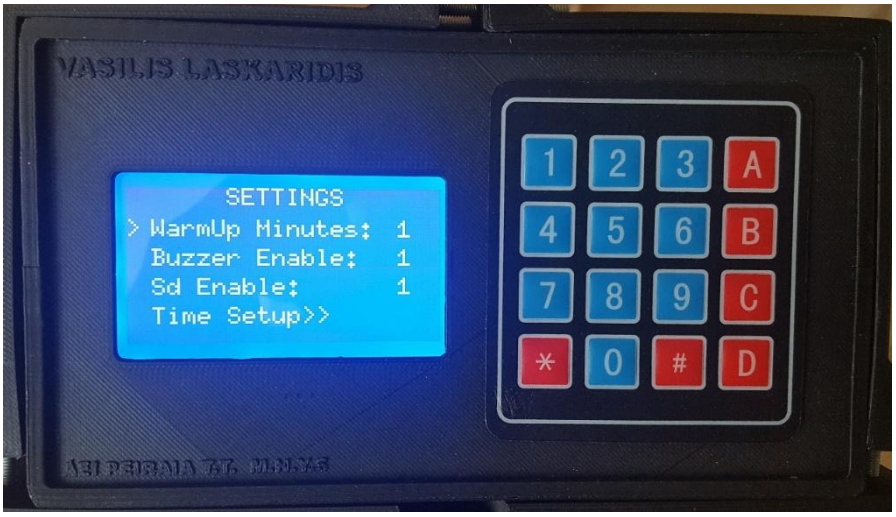
Εικόνα 5.2. Μενού Εισαγωγής κωδικού ασθενή

Μενού Ρυθμίσεων Μικροϋπολογιστή

Σε αυτό το μενού δίνεται η δυνατότητα να μεταβληθούν οι αρχικές ρυθμίσεις του μικροϋπολογιστή. Για να επιλέξουμε το πεδίο που επιθυμούμε, πατάμε το πλήκτρο «2» για να ανεβούμε και το πλήκτρο «8» για να κατεβούμε στο μενού.

Αφού περιηγηθούμε στο μενού, επιλέγουμε το επιθυμητό πεδίο και πατώντας το πλήκτρο «A», μεταβάλλεται ο χρόνος προθέρμανσης π.χ warmup minutes από 1 σε 2, 3, κ.λ.π. Για τα πεδία Buzzer Enable και Sd Enable το “1” αντιστοιχεί σε ΕΝΕΡΓΟ και το “0” σε ΑΝΕΝΕΡΓΟ.

Σε περίπτωση που επιλεγεί το πεδίο «Time Setup», αυτόματα εμφανίζεται η οθόνη του μενού ρύθμισης ώρας και ημερομηνίας. Αφού ρυθμίσουμε την ώρα και την ημερομηνία, πατώντας το πλήκτρο «B» επιστρέφουμε στο κεντρικό μενού



Εικόνα 5.3. Μενού Ρυθμίσεων Μικροϋπολογιστή

Μενού Ρύθμισης Ώρας και Ημερομηνίας

Η ώρα και η ημερομηνία ορίζεται πιέζοντας τα αριθμητικά πλήκτρα. Πρέπει να ληφθεί υπόψη ότι για να δεχθεί νέα δεδομένα το πεδίο πρέπει να είναι κενό. Αν το πεδίο περιέχει δεδομένα καθαρίζονται πατώντας το πλήκτρο «C».

Για να περιηγηθούμε στο μενού πατάμε το πλήκτρο «A» για το επόμενο πεδίο και το πλήκτρο «B» για το προηγούμενο πεδίο.

Εικόνα 5.4. Μενού Ρύθμισης Ώρας και Ημερομηνίας

Μενού προβολής δεδομένων ασθενούς

Στο μενού αυτό αναγράφονται οι επιτευχθείσες μέγιστες και ελάχιστες μοίρες κάθε άρθρωσης του συγκεκριμένου ασθενούς με τον ίδιο τρόπο, όπως στο κεντρικό μενού. Αφού ολοκληρώσουμε την καταγραφή επιστρέφουμε στο κεντρικό μενού, πιέζοντας το πλήκτρο «B».

Εικόνα 5.5. Μενού προβολής δεδομένων ασθενούς

ΚΕΦΑΛΑΙΟ 6. ΜΗΧΑΝΟΛΟΓΙΚΟΣ ΣΧΕΔΙΑΣΜΟΣ

Σε αυτό το κεφάλαιο θα αναλυθούν η μηχανική δομή και τα συστατικά μέρη του οργάνου εκγύμνασης.

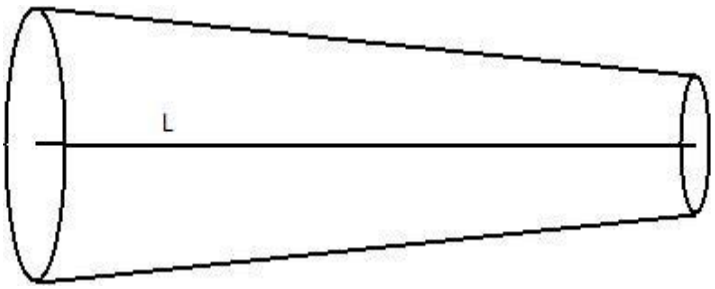
6.1. Σχεδιαστική μελέτη

Στο στάδιο της σχεδίασης της πλατφόρμας υπολογίστηκαν οι δυνάμεις και οι ροπές που ασκούνται στις διάφορες συνδέσεις της κατασκευής, διότι αυτές είναι τα αδύναμα σημεία από μηχανική άποψη.

Για να εξασφαλίσουμε την απρόσκοπτη λειτουργία της πλατφόρμας στην διάρκεια του χρόνου, αλλά και την ασφάλεια του ασθενή πρέπει να ξέρουμε τις μέγιστες τιμές τους.

Πειραματικά δοκιμάστηκε η αντοχή των συνδέσεων εφαρμόζοντας δυνάμεις διπλάσιες των υπολογιζόμενων μέγιστων τιμών. Δεν παρατηρήθηκε καμία μεταβολή στις ηλεκτροκόλλησεις των συνδέσεων. Παραθέτονται λεπτομερώς οι υπολογισμοί για την ροπή της δύναμης που ασκείται από το βάρος του οργάνου - Mo και του βάρος του χεριού του ασθενή - Mp, πάνω στην κλείδωση της πλάτης. Οι άλλες ροπές υπολογίζονται με παρόμοιο τρόπο. Βασικές παραδοχές που χρησιμοποιούνται είναι ότι έχουμε:

1. Ομοιόμορφη κατανομή της πυκνότητας του ιστού του ανθρώπινου χεριού.
2. Το ανθρώπινο χέρι γεωμετρικά είναι ένας κώνος.



Εικόνα 6.1. Γεωμετρική αναπαράσταση ανθρώπινου χεριού

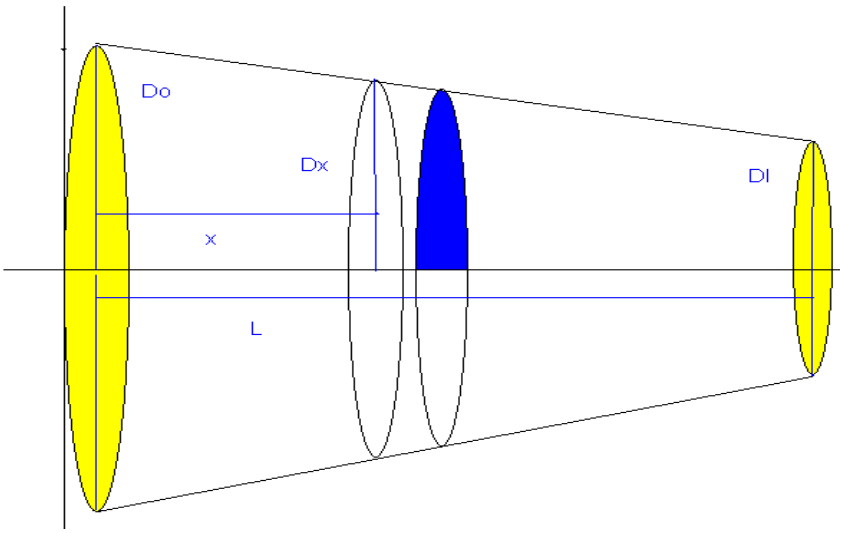
Όπως ξέρουμε η ροπή της δύναμης ως πως κάποιο σημείο, δίνεται από τον τύπο:

$$\vec{M} = \vec{r} \times \vec{F}.$$

Αφού χωρίζουμε το χέρι σε άπειρες τομές, όπως φαίνεται στην εικόνα 6.2, παρατηρούμε ότι η στοιχειώδης μάζα κάθε τομής, σε απόσταση x από την βάση του κώνου είναι dm = $\rho . S(x) . dx$, όπου S(x) το εμβαδών τις τομής.

Άρα η στοιχειώδεις ροπή της δύναμης της βαρύτητας του χεριού είναι:

$$dM_x = g * dm * x = g * \rho * S(x) * x * dx$$



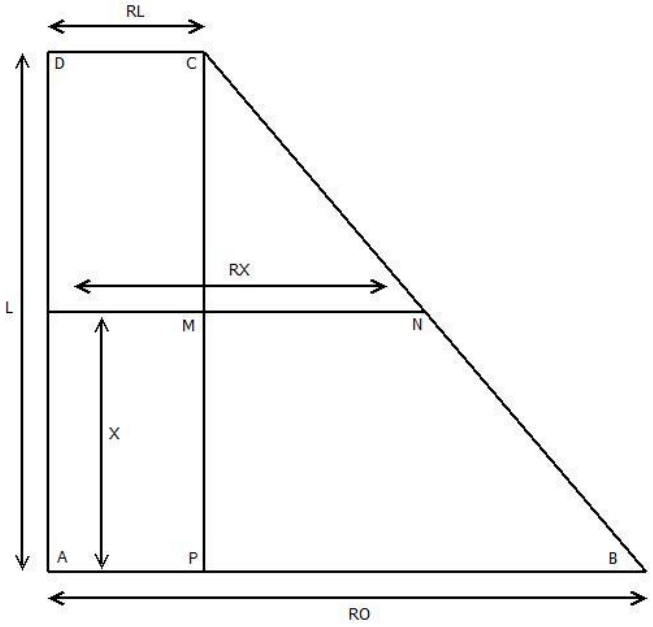
Εικόνα 6.2. Γεωμετρικές τομές ανθρώπινου χεριού

Η ολική ροπή είναι το άθροισμα όλων των στοιχειωδών ροπών:

$$M = \int_0^L dM_x = \int_0^L g * \rho * S(x) * x * dx = \rho * g * \int_0^L S(x) * x * dx \text{ [1]}$$

Για να υπολογιστεί το ολοκλήρωμα πρέπει να ξέρουμε το εμβαδόν της τομής, ως συνάρτηση της απόστασης x, δηλαδή να βρούμε μια αναλυτική της έκφραση. Εδώ κάνουμε μια εγκάρσια τομή του κώνου, όπως φαίνεται στην εικόνα 6.2.

Στο σχήμα RO, είναι η ακτίνα του χεριού στο ύψος του ωμού.
RL είναι η ακτίνα τις τομής στο ύψος του καρπού.
Ενώ RX είναι η ακτίνα σε απόσταση X.



Εικόνα 6.3. Κατά μήκος τομή κώνου

Από τα όμοια τρίγωνα CPB ≈CMN έχουμε:

$\frac{R_x - R_L}{R_o - R_L} = \frac{L-x}{L}$, $R_x = R_o - \frac{x}{L}(R_o - R_L) = R_o - \frac{x.\Delta R}{L}$, ΔR= $R_o - R_L$

$S(x) = \pi * R_x^2 = \pi * \left(R_o - \frac{x.\Delta R}{L}\right)^2 = \pi * (R_o^2 - 2 * \frac{\Delta R.R_o}{L} * x + \frac{\Delta R^2.x^2}{L^2})$.

Αντικαθιστώντας S(x) με την παραπάνω αναλυτική της έκφραση στον τύπο [1]

Για την ολική ροπή έχουμε:

$Mp = \pi * g * \rho * \int_0^L \left(R_o^2 - 2 * \frac{\Delta R.R_o}{L} * x + \frac{\Delta R^2.x^2}{L^2}\right) * x * dx =$

$= \pi * g * \rho * [R_o^2 * \int_0^L x dx - 2 * \frac{\Delta R.R_o}{L} * \int_0^L x^2 \, dx + \frac{\Delta R^2}{L^2} * \int_0^L x^3 \, dx]$

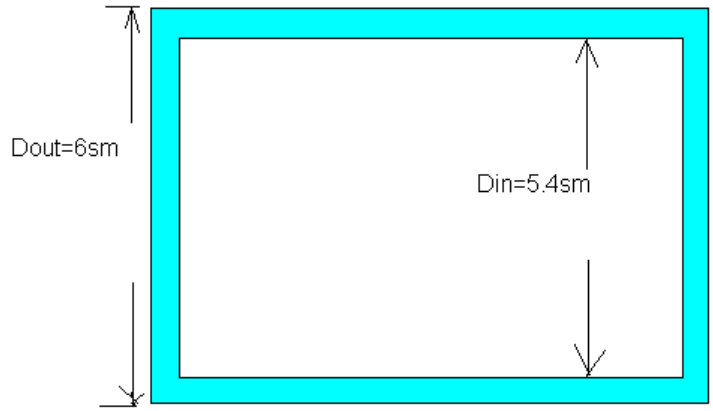
$= \pi * g * \rho * [\frac{R_o^2}{2} * L^2 - 2 * \frac{\Delta R.R_o}{L} * \frac{L^3}{3} + \frac{\Delta R^2}{L^2} * \frac{L^4}{4}] = \pi * g * \rho * \frac{L^2}{2} * [R_o^2 - \frac{4.\Delta R.R_o}{3} + \frac{\Delta R^2}{2}]$

Ακόμη πιο ευκολά υπολογίζεται η ροπή Mo.

Στο σχήμα (6.4) έχουμε την εγκάρσια τομή του κοιλοδοκού.

D_{out} - Εξωτερική διάσταση

D_{in} - Εσωτερική διάσταση



Εικόνα 6.4. Εγκάρσια τομή κοιλοδοκού.

Mo= $g * \rho_o * \int_0^L S(x) * x * dx = g * \rho_o * \int_0^L (D_{out}^2 - D_{in}^2) * x * dx = g * \rho_0 * (D_{out}^2 - D_{in}^2) * L^2/2$

Η ολική ροπή είναι το άθροισμα M=M0 +Mp,

Για να έχουμε μια ρεαλιστική αντίληψη για το ποιες είναι οι μέγιστες τιμές των ροπών που μπορεί να προκύψουν, αντικαθιστούμε τους παραπάνω τύπους με τις παρακάτω τιμές για έναν φυσιολογικό άνθρωπο.

L =0,5m, το μήκος του χεριού

R0 = 0.05 sm , η ακτίνα του χεριού στο ύψος της άνω κλείδωσης

RL = 0,01 sm , η ακτίνα του χεριού στο επίπεδο του καρπού

g=10m/s2, η βαρυτική επιτάχυνση

ρ=1000kg/m3, η πυκνότητα του ιστού του ανθρώπινου χεριού

ρ0=8000kg/m3, πυκνότητα του ανοξείδωτου χάλυβα

Τότε ,

Mp= 3,14*10*1000*0,5*2*((0,1)*2- (4*(0,05)*0,1)/3 + 0,026)/4 [N.m]

= 11.5 [N.m]

M0 = 10*8000*(0,5)*2*((0,06)*2 – 0,027) [N.m]

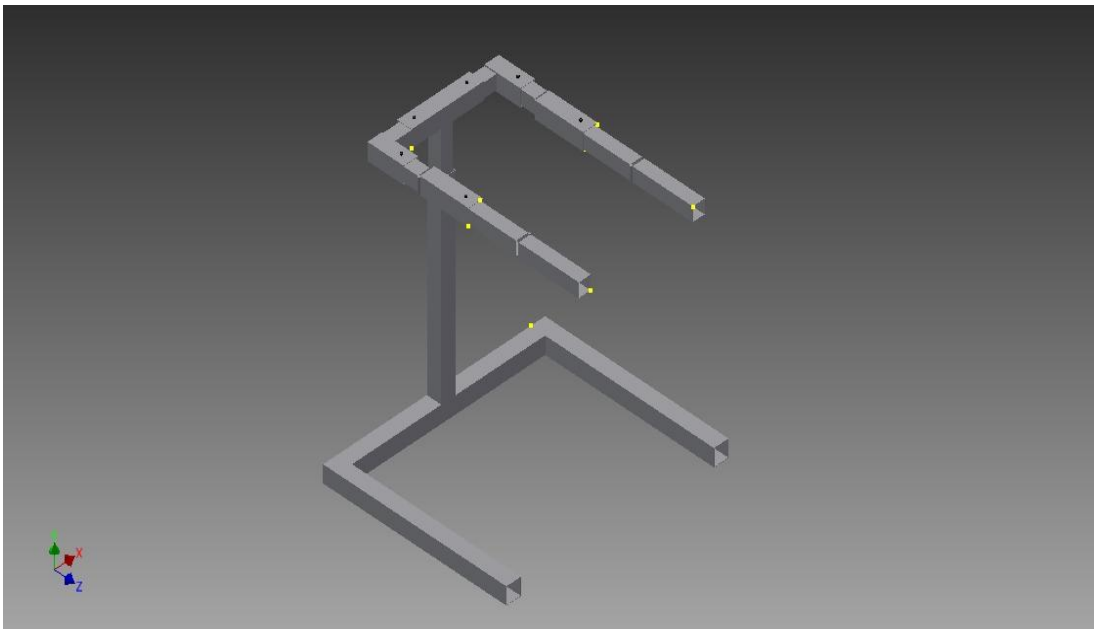
= 13,7 [N.m]

M ≈ 25 [N.m]

6.2. Βασικά μέρη πλατφόρμας εκγύμνασης

Αυτή η ενότητα περιγράφει την κατασκευή του μηχανικού μέρους της πλατφόρμας εκγύμνασης. Μία ισομετρική όψη ενός στερεού μοντέλου της παρουσιάζεται στην εικόνα 6.6.

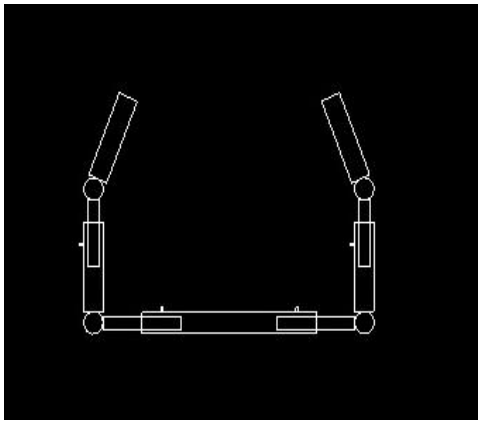
Το μοντέλο σχεδιάστηκε με τη χρήση του λογισμικού πακέτου Autodesk Inventor.



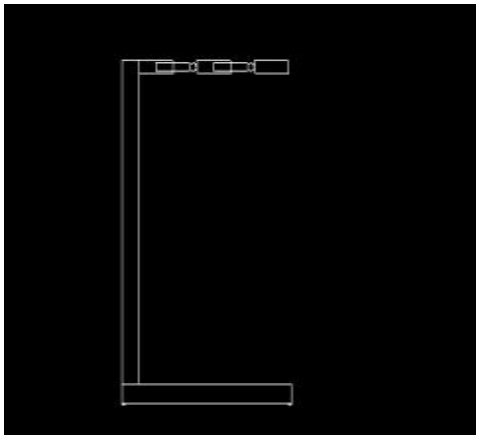
Εικόνα 6.5. Μοντέλο πλατφόρμας

Η διάταξη της πλατφόρμας κατασκευάστηκε ώστε να μπορεί να προσαρμοστεί εύκολα σε κάθε σωματότυπο. Για αυτό το λόγο μπορούν να μεταβάλλονται οι διαστάσεις της πλάτης και των χεριών, δηλαδή η απόσταση του άξονα από τον αυχένα και τέλος το ύψος της πλάτης (βλ. εικόνες 6.6, 6.7.

Η πλατφόρμα σχεδιάστηκε με σκοπό να μπορεί να προσαρμόζεται σε κάθε χρήστη, που χρησιμοποιεί αναπηρικό καροτσάκι η όχι και εύκολα να μπορεί να ενσωματωθεί καρέκλα, στη βάση της πλατφόρμας.



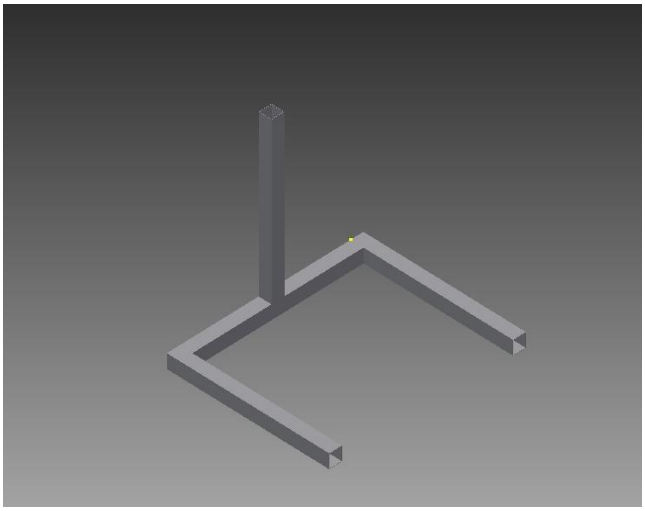
Εικόνα 6.6. Κάτω όψη πλατφόρμας



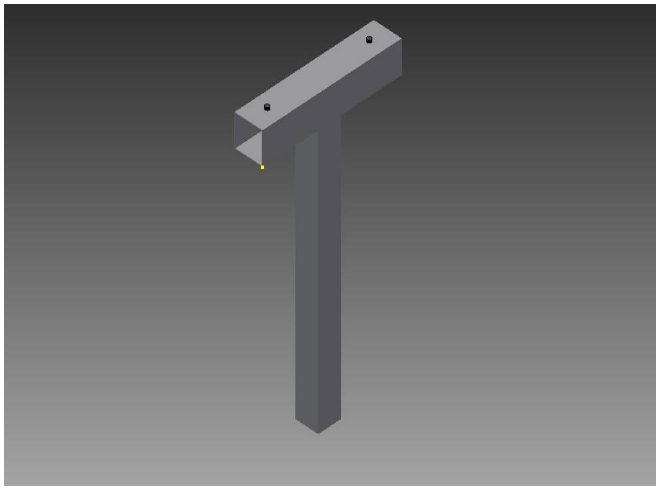
Εικόνα 6.7. Πλαγιά όψη πλατφόρμας

Το συνολικό ύψος της πλατφόρμας είναι (min 820 , max 1100) χιλιοστά, το πλάτος είναι (850) χιλιοστά και το μήκος είναι (755) χιλιοστά. Η πλατφόρμα θεραπεία αποτελείται από πέντε βασικά υποσυγκροτήματα:

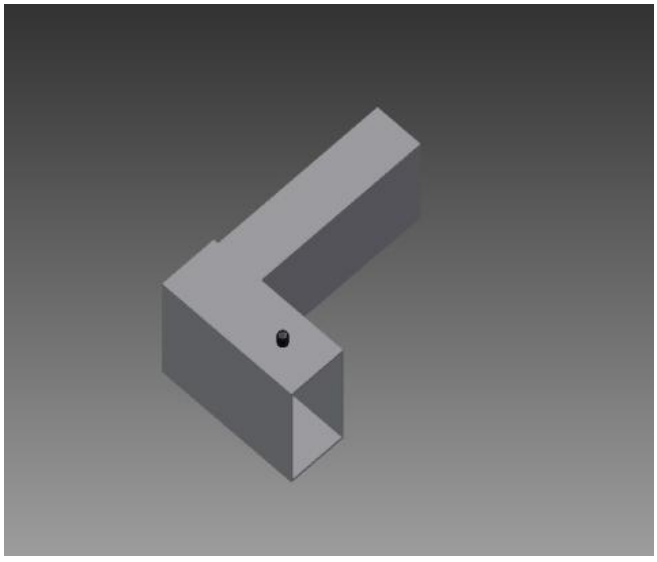
1. Βάση (βλέπε Εικόνα 6.8.),
2. Πλάτη (βλέπε Εικόνα 6.9.),
3. Αριστερός και δεξιός ώμος (βλέπε Εικόνα 6.10.),
4. Αριστερό και δεξιό μπράτσο (βλέπε Εικόνα 6.11.),
5. Αριστερό και δεξιό πήχη (βλέπε Εικόνα 6.12.),



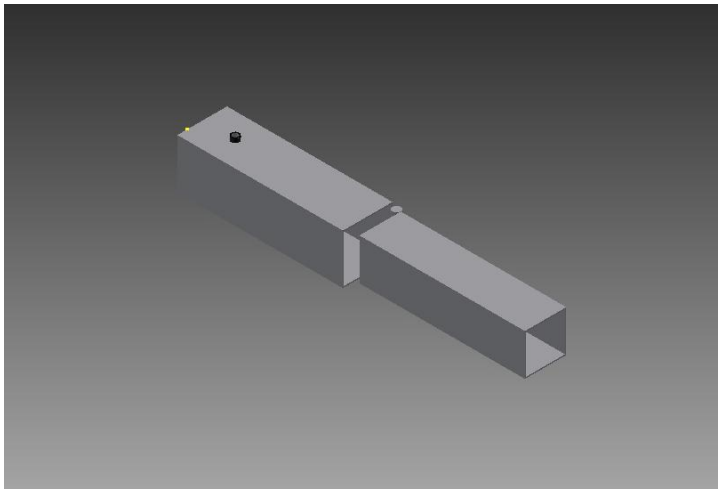
Εικόνα 6.8. Βάση πλατφόρμας



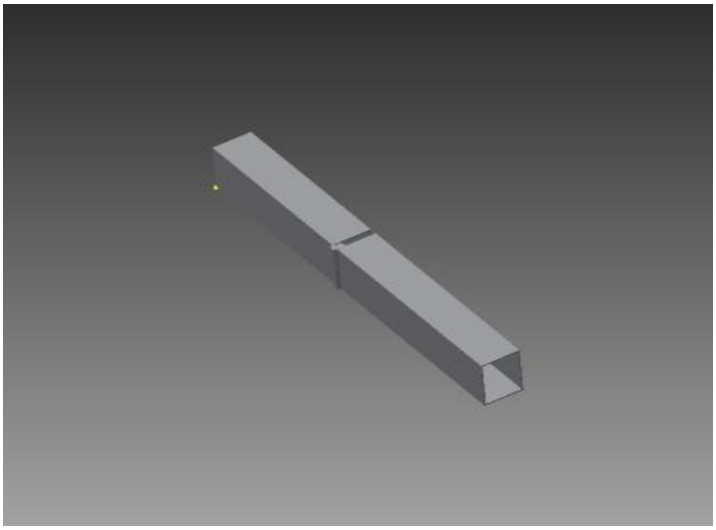
Εικόνα 6.9. Πλάτη πλατφόρμας



Εικόνα 6.10. Αριστερός και δεξιός ώμος πλατφόρμας



Εικόνα 6.11. Αριστερό και δεξιό μπράτσο πλατφόρμας



Εικόνα 6.12. Αριστερό και δεξιό πήχης πλατφόρμας

Τα μηχανικά μέρη του οργάνου είναι κατασκευασμένα από τρεις κοιλοδοκούς βάσης, ενωμένοι μεταξύ τους σε σχήμα Π, μήκους 43 cm και πάχους 6 cm. Στο κέντρο του μεσαίου κοιλοδοκού βρίσκεται ένας κάθετος κοιλοδοκός, ύψους 75 cm. Στον μεσαίο κοιλοδοκό εισέρχεται ένας κοιλοδοκός σχήματος «**Τ»** κυμαινόμενου ύψους 83-110 cm και μήκους 30 cm, ιδίων διαστάσεων, ο οποίος συνδέεται δεξιά και αριστερά με κοιλοδοκούς (βραχίονες) σε σχήμα «**Γ»** εκατέρωθεν, κυμαινόμενου μήκος 80-93 cm και κυμαινόμενου πλάτους 26-35 cm.

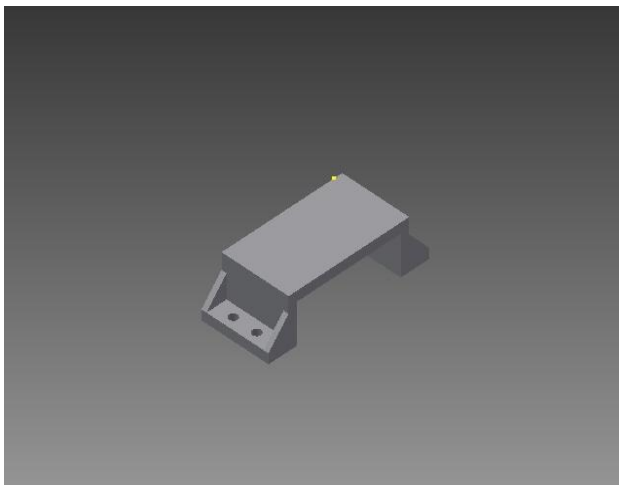
Οι διακυμάνσεις των κοιλοδοκών επιτυγχάνονται με στρόφιγγες, οι οποίες προσαρμόζονται στις εκάστοτε διαστάσεις των ασθενών. Το όργανο εκγύμνασης φέρει συνολικά 7 στρόφιγγες.

6.3. Επιμέρους εξαρτήματα πλατφόρμας εκγύμνασης

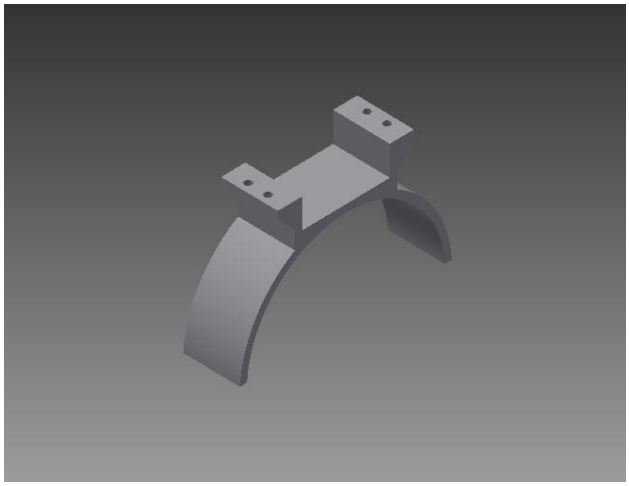
Σε αυτή την ενότητα περιγράφεται η σχεδίαση των επιμέρους εξαρτημάτων του οργάνου εκγύμνασης.

6.3.1. Βάση στήριξης χεριού

Αρχικά σχεδιάστηκε βάση στην οποία στηρίζεται το χέρι του ασθενή πάνω στο όργανο εκγύμνασης. Ένας ιμάντας Velcro χρησιμοποιείται για τη σωστή στερέωση του κάθε χεριού στο όργανο εκγύμνασης.



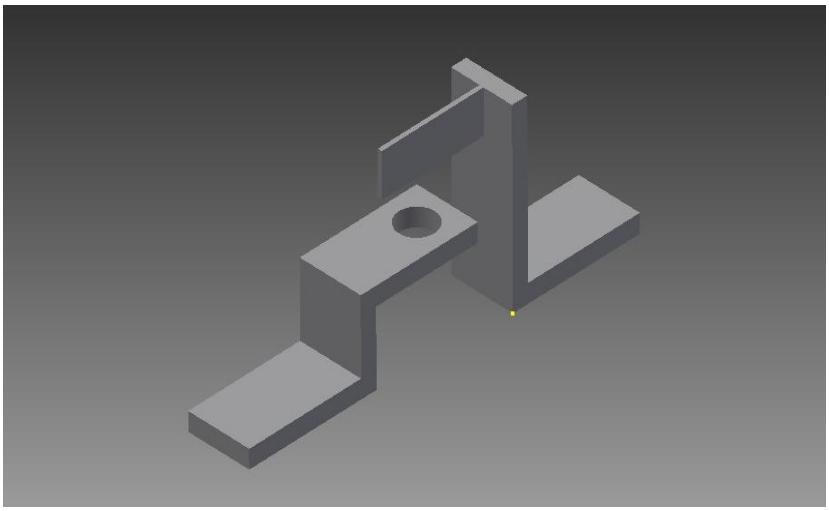
Εικόνα 6.13. Σφικτήρας βάσης χεριού



Εικόνα 6.14. Βάση στήριξης χεριού

6.3.2. Βάση στήριξης ποτενσιόμετρων

Οι βάσεις στήριξης των ποτενσιομέτρων είναι (4) τέσσερεις, μία για κάθε κλείδωση και σχεδιάστηκαν με σκοπό να στερεώνουν το ποτενσιόμετρο πάνω στον άξονα κλείδωσης (μεντεσές), ώστε να καταμετρούνται οι μοίρες κάθε άρθρωσης. Η σχεδίαση εστιάστηκε στην απορρόφηση των ανοχών του μεντεσέ, για την αποφυγή της φθοράς των ποτενσιομέτρων.

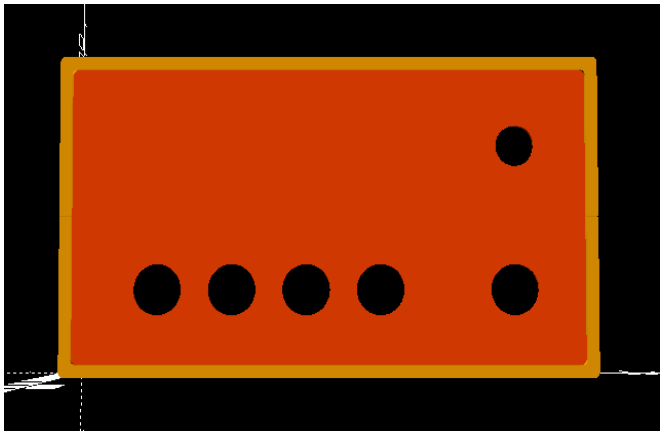


Εικόνα 6.15. Βάση στήριξης ποτενσιομέτρων

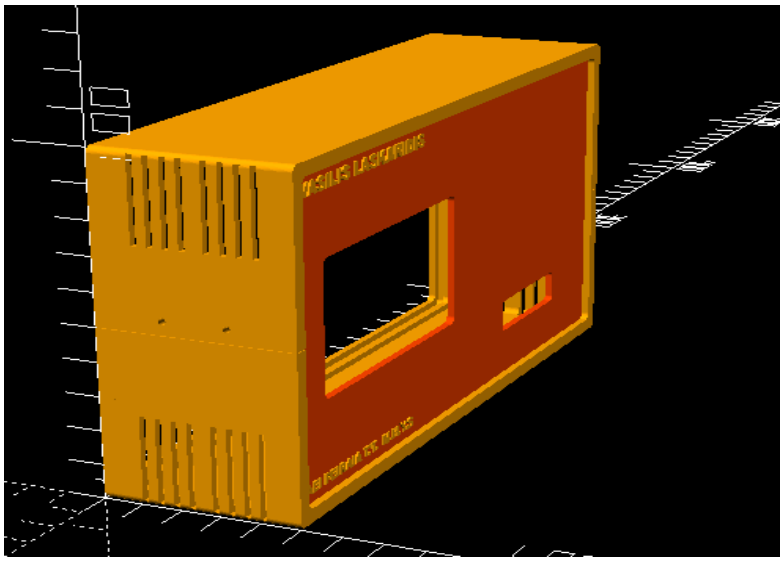
6.3.3. Σχεδιασμός περιβλήματος ηλεκτρονικών εξαρτημάτων

Το κουτί των ηλεκτρονικών εξαρτημάτων έχει διαστάσεις 50 x 180 x 100 mm και σχεδιάστηκε με την βοήθεια του προγράμματος open SCAD. Η σχεδίαση βασίστηκε σε πηγαίο κώδικα (ο κώδικας περιλαμβάνεται στο παράρτημα). Το κουτί φέρει υποδοχές για την οθόνη, το πληκτρολόγιο, τα ποτενσιόμετρα, την τροφοδοσία καθώς και ανάγλυφη επιγραφή με το όνομα του συγγραφέως και της σχολής του.

Εικόνα 6.16. Μπροστά όψη περιβλήματος



Εικόνα 6.17. Πίσω όψη περιβλήματος



Εικόνα 6.18. Πλάγια όψη περιβλήματος

7. ΣΥΜΠΕΡΑΣΜΑΤΑ

Ολοκληρώνοντας την υλοποίηση της παρούσας πτυχιακής εργασίας, γίνονται κατανοητές οι δυσκολίες μιας τόσο απαιτητικής κατασκευής, αλλά και τα προτερήματα της.

Πλεονεκτήματα για τον ασθενή:

- Διευκολύνει την άσκηση λόγω της απόσβεσης της βαρύτητας των άνω άκρων, χωρίς καμία καταπόνηση των αρθρώσεων του ασθενούς.
- Παρουσιάζει στον ασθενή το score κάθε συνεδρίας, μέσω του μικροϋπολογιστή που διαθέτει.
- Σχεδιασμένο να κρατάει ενεργό και να παρακινεί τον ασθενή για εκπαίδευση, ενθαρρύνοντας τον να συνεχίσει ακούγοντας ηχητικό μήνυμα που αφορά την κατάρριψη του προηγούμενου score από το νέο, το οποίο επιτυγχάνεται μετρώντας τις μοίρες της άρθρωσης που γυμνάζονται.
- Ενισχύει το κίνητρο του ασθενούς για αποκατάσταση. Συστατικό, άκρως απαραίτητο στους ηλικιωμένους ασθενείς, που συχνά παραιτούνται λόγω της αργής και επίπονης θεραπείας.
- Εμφανίζει τα επιτευχθέντα δεδομένα εκάστης συνεδρίας και του συνόλου αυτών στον φυσικοθεραπευτή, ώστε να είναι εφικτή η έκδοση στατιστικών δεδομένων και ο προγραμματισμός των μελλοντικών προγραμμάτων άσκησης.
- Ασφαλές περιβάλλον για να ενισχύει την αποκατάσταση-θεραπεία,
- Ειδικοί αισθητήρες ελέγχουν όλη την κίνηση του ασθενή.
- Δυνατότητα αναβάθμισης της πλατφόρμας προκειμένου να παρέχει δεδομένα, που αφορούν τους παλμούς τις καρδιάς του ασθενούς κατά την εκπόνηση της άσκησης,
- Δυνατότητα προσθήκης συσκευής υποβοήθησης, σε πολύ βαριά περιστατικά.

Πλεονεκτήματα για τον θεραπευτή:

- Έξυπνο και απλό περιβάλλον εργασίας για τον χρήστη.
- Πλήρης έλεγχος στην προσαρμογή ασκήσεων ειδικά για τον κάθε ασθενή.
- Συλλογή και διαχείριση δεδομένων και εξαγωγή Reports της θεραπείας και της προόδου του ασθενή.
- Λεπτομερής ανάλυση της προόδου του ασθενούς.
- Παρακολούθηση και καταγραφή της κίνησης.
- Δυνατότητα εκτύπωσης-εξαγωγής των Reports με γραφήματα.

Για τις Κλινικές Αποκατάστασης:

- Χαμηλό κόστος κατασκευής οργάνου εκγύμνασης.
- Καθιστά πιο οικονομική τη χορήγηση θεραπείας επαναληπτικής κίνησης,
- Αυξάνει τη ροή των ασθενών

Εν κατακλείδι, στην παρούσα πτυχιακή εργασία περιγράψαμε διεξοδικά τα επιμέρους στοιχεία που αποτελούν την πειραματική διάταξη και εξηγήθηκε ο τρόπος με τον οποίο κατασκευάστηκαν τα εξαρτήματα για την δημιουργία ενός οργάνου αποκατάστασης άνω ακρών.

Κατά την εκπόνηση της εργασίας έγιναν δοκιμές με βηματική μεταβολή φορτίου στους δύο βασικούς άξονες του οργάνου. Σε κάθε περίπτωση τα πειραματικά αποτελέσματα κρίνονται ικανοποιητικά, καθιστώντας την λειτουργία του οργάνου αποδοτική και ασφαλή για τον ασθενή.

Το μεγάλο συγκριτικό πλεονέκτημα του οργάνου είναι το χαμηλό κόστος κατασκευής, σε σχέση με τα υπόλοιπα όργανα αποκατάστασης της αγοράς.

Θεωρώ δεδομένο ότι η πλατφόρμα αποκατάστασης άνω άκρων, θα αναβαθμίσει τον τρόπο αποθεραπείας των ασθενών που πάσχουν από εγκεφαλική παράλυση, και νοσηλεύονται στο Εθνικό Κέντρο Αποκατάστασης.

8. ΠΑΡΑΡΤΗΜΑ Α΄

ΠΗΓΑΙΟΣ ΚΩΔΙΚΑΣ ΟΡΓΑΝΟΥ ΕΚΓΥΜΝΑΣΗΣ

```
//////////////////////////////////////////Libraries//////////////////////////////////////////
#include "U8glib.h"
#include <Keypad.h>
#include <SD.h>
#include <Wire.h>
#include <RealTimeClockDS1307.h>
#include <EEPROM.h>
//////////////////////////////////////////Libraries//////////////////////////////////////////

//////////////////////////////////////////SD/File Variable//////////////////////////////////////////
File myFile;//File variable to save excel file
const int chipSelectpin = 10;//The CS pin of SD car
String userwithpath;//Variable to storage the file name with the excel ending(.csv)
char filename[10];//Variable to storage the file name of the excel
String user ;//Variable to storage the username from the keypad
//////////////////////////////////////////SD/File Variable//////////////////////////////////////////

//////////////////////////////////////////Screen Variable//////////////////////////////////////////
U8GLIB_ST7920_128X64_4X u8g(6, 7, 8);  // SPI Connection of the screen: SCK = en =
18, MOSI = rw = 16, CS = di = 17
//////////////////////////////////////////Screen Variable//////////////////////////////////////////

//////////////////////////////////////////Keypad Variable//////////////////////////////////////////
const byte ROWS = 4;//Number of row
const byte COLS = 4;//Number of column
char keys[ROWS][COLS] = {//The array of keypad button
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
byte rowPins[ROWS] = {37, 35, 33, 31}; //The row pinouts of the keypad
byte colPins[COLS] = {29, 27, 25, 23}; //The column pinouts of the keypad
Keypad Kpd = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS);
```

```
char Key;//Keypad output
//////////////////////////////////////Keypad Variable//////////////////////////////////////

//////////////////////////////////////PinOuts//////////////////////////////////////
const int buzzerpin = 9;
//////////////////////////////////////PinOuts//////////////////////////////////////

//////////////////////////////////Potentiometer Sensor Pins//////////////////////////////////
const int pot1 = A0;//Potentiometer sensor of right shoulder
const int pot2 = A1;//Potentiometer sensor of right elbow
const int pot3 = A2;//Potentiometer sensor of left shoulder
const int pot4 = A3;//Potentiometer sensor of left elbow
//////////////////////////////////Potentiometer Sensor Pins//////////////////////////////////

////////////////////////////////Potentiometer Sensor Variables////////////////////////////////
int pot1value;//Degrees of the sensor
int pot2value;
int pot3value;
int pot4value;
int pot1maxvalue;//Max degrees of the sensor
int pot2maxvalue;
int pot3maxvalue;
int pot4maxvalue;
int pot1minvalue;//Min degrees of the sensor
int pot2minvalue;
int pot3minvalue;
int pot4minvalue;
////////////////////////////////Potentiometer Sensor Variables////////////////////////////////

//////////////////////////////////////Score Variables//////////////////////////////////////
int score = 0;//Variable to storage the score
int lastscore = 0;//Variable to storage the last score
//////////////////////////////////////Score Variables//////////////////////////////////////

//////////////////////////////////////Program Variables//////////////////////////////////////
unsigned long int userworktime;//Times counter for warm-up function
byte startwork = 0;//Warm-up Variable
byte warmup = 0;//Warm-up Variable
byte finishwarmup = 0;//Warm-up Variable
int screen = 0;//Screen Variable
```

```
int settingsmenu = 0;//Cursor Variable of settings menu
int Rtcscreen = 0;//Real time clock setup screen variable
//////////////////////////////////////////Program Variables//////////////////////////////////////////

//////////////////////////////////////////Settings Variables//////////////////////////////////////////
byte BuzzerEnable ;//Variable to enable or disable the buzzer
byte SdWriteEnable;//Variable to enable or disable the SD write
byte warmuptime;//Minutes of the warm-up
//////////////////////////////////////////Settings Variables//////////////////////////////////////////

////////////////////////////////////Real Time Clock Variables////////////////////////////////////
String datesetup, monthsetup, yearsetup, hoursetup, minutessetup;
////////////////////////////////////Real Time Clock Variables////////////////////////////////////



void setup() {
  Serial.begin(115200);
  pinMode(buzzerpin, OUTPUT);
  pinMode(chipSelectpin, OUTPUT);
  pinMode(buzzerpin, OUTPUT);
  pinMode(pot1, INPUT);
  pinMode(pot2, INPUT);
  pinMode(pot3, INPUT);
  pinMode(pot4, INPUT);
  if ( u8g.getMode() == U8G_MODE_R3G3B2 )u8g.setColorIndex(255);
  else if ( u8g.getMode() == U8G_MODE_GRAY2BIT ) u8g.setColorIndex(3);
  else if ( u8g.getMode() == U8G_MODE_BW ) u8g.setColorIndex(1);
  else if ( u8g.getMode() == U8G_MODE_HICOLOR ) u8g.setHiColorByRGB(255, 255,
255);
  warmuptime = EEPROM.read(0);
  BuzzerEnable = EEPROM.read(1);
  SdWriteEnable = EEPROM.read(2);
  if (!SD.begin(chipSelectpin)) {
    Serial.println("SD Initialization Faild!");
    screen = 4;
  } else Serial.println("SD Initialization Done.");
}
```

```
void loop() {
  //Switch between Screens
  //screen=0 Main
  //screen=1 Insertuser
  //screen=2 Datalogger
  //screen=3 Settings
  //screen=4 Sderror
  //screen=5 RtcSetup

  u8g.firstPage();
  do {
    if (screen == 0)mainscreen();
    if (screen == 1)insertuserscreen();
    if (screen == 2)dataloggerscreen();
    if (screen == 3)settingsscreen();
    if (screen == 4)sderrorscreen();
    if (screen == 5)RtcSetupScreen();
  } while (u8g.nextPage());
  if (screen == 0)mains();
  if (screen == 1)insertuser();
  if (screen == 2)datalogger();
  if (screen == 3)settings();
  if (screen == 4)sderror();
  if (screen == 5)RtcSetup();
}




void mains() {
  pot1value = map(analogRead(pot1), 0, 1023, 0, 270);
  pot2value = map(analogRead(pot2), 0, 1023, 0, 270);
  pot3value = map(analogRead(pot3), 0, 1023, 0, 270);
  pot4value = map(analogRead(pot4), 0, 1023, 0, 270);

  if (pot1value >= pot1maxvalue)pot1maxvalue = pot1value;
```

```
  if (pot1value <= pot1minvalue)pot1minvalue = pot1value;
  if (pot2value >= pot2maxvalue)pot2maxvalue = pot2value;
  if (pot2value <= pot2minvalue)pot2minvalue = pot2value;
  if (pot3value >= pot3maxvalue)pot3maxvalue = pot3value;
  if (pot3value <= pot3minvalue)pot3minvalue = pot3value;
  if (pot4value >= pot4maxvalue)pot4maxvalue = pot4value;
  if (pot4value <= pot4minvalue)pot4minvalue = pot4value;

  if (user == "") {
    userworktime = 0;
    startwork = 0;
    score = 0;
    lastscore = 0;
    warmup = 0;
    finishwarmup = 0;
    pot1maxvalue = 0;
    pot1minvalue = 0;
    pot2maxvalue = 0;
    pot2minvalue = 0;
    pot3maxvalue = 0;
    pot3minvalue = 0;
    pot4maxvalue = 0;
    pot4minvalue = 0;
  } else if (user.length() == 5 && startwork == 0) {
    userworktime = millis();
    pot1minvalue = pot1value;
    pot2minvalue = pot2value;
    pot3minvalue = pot3value;
    pot4minvalue = pot4value;
    startwork = 1;
    warmup = 1;
    finishwarmup = 1;
  }

  if (userworktime + (warmuptime * 60000) <= millis()) {//Ckeck if user finish warm-up
    warmup = 0;
    if (finishwarmup == 1) {//When warm-up finished clear the potentiometer
variable and call WriteToSd to write score on sd.
      finishwarmup = 0;
      if (SdWriteEnable == 1)WriteToSd(1);
```

```
      pot1minvalue = pot1value;
      pot2minvalue = pot2value;
      pot3minvalue = pot3value;
      pot4minvalue = pot4value;
      pot1maxvalue = pot1value;
      pot2maxvalue = pot2value;
      pot3maxvalue = pot3value;
      pot4maxvalue = pot4value;
    }

    if (user.length() == 5)score = (pot1maxvalue - pot1minvalue) + (pot2maxvalue -
pot2minvalue) + (pot3maxvalue - pot3minvalue) + (pot4maxvalue -
pot4minvalue);//calculate the score after warm-up

    if (score > lastscore) {//if the Score is bigger than previous Score reward user
      if (SdWriteEnable == 1)WriteToSd(0);//Call WriteToSd function to write new score to
excel
      if (BuzzerEnable == 1) {
        digitalWrite(buzzerpin, HIGH);
        delay(50);
        digitalWrite(buzzerpin, LOW);
        delay(72);
        digitalWrite(buzzerpin, HIGH);
        delay(50);
        digitalWrite(buzzerpin, LOW);
      }
      lastscore = score;
    }

  } else if (user.length() == 5)lastscore = (pot1maxvalue - pot1minvalue) +
(pot2maxvalue - pot2minvalue) + (pot3maxvalue - pot3minvalue) + (pot4maxvalue -
pot4minvalue);//Calculate the score in the warm up

  Key = Kpd.getKey();//Check if button is pressed
  if (Key) {
    if (Key == 'B') user = "";//If "B" is pressed clear user
    if (Key == 'A') {//If "A" is pressed go to Insertuser screen
      if (user == "")screen = 1;
      else Serial.println("Another user is running");
    }
```

```
    if (Key == 'C')screen = 3;//If "C" is pressed go to Settings screen
    if (Key == 'D')screen = 2;//If "D" is pressed go to Datalogger screen
  }

}

void mainscreen() {
  u8g.setFont(u8g_font_helvB12);//Set the Font of the Screen

  u8g.setPrintPos(0, 12);//Print potentiometer degrees
  u8g.print(String(pot4value));

  u8g.setPrintPos(0, 64);//Print potentiometer degrees
  u8g.print(String(pot3value));

  if (pot2value >= 100)u8g.setPrintPos(102, 12);//Print potentiometer degrees
  else if (pot2value >= 10)u8g.setPrintPos(111, 12);
  else if (pot2value >= 0)u8g.setPrintPos(120, 12);
  u8g.print(String(pot2value));

  if (pot1value >= 100)u8g.setPrintPos(102, 64);//Print potentiometer degrees
  else if (pot1value >= 10)u8g.setPrintPos(111, 64);
  else if (pot1value >= 0)u8g.setPrintPos(120, 64);
  u8g.print(String(pot1value));

  if (score >= 1000)u8g.setPrintPos(46, 38); //Print potentiometer degrees
  else if (score >= 100)u8g.setPrintPos(51, 38);
  else if (score >= 10)u8g.setPrintPos(55, 38);
  else if (score >= 0)u8g.setPrintPos(60, 38);
  u8g.print(String(score));
  u8g.setFont(u8g_font_6x10);//Set the Font of the Screen

  u8g.setPrintPos(50, 24); //Print "Score" label and username
  u8g.print("Score");
  if (user == "") {
    u8g.setPrintPos(43, 7);
    u8g.print("No user");
  } else if (user.length() == 5) {
    u8g.setPrintPos(50, 7);
    u8g.print(user);
```

```
  }

  if (warmup == 1) {//If its warm-up time print "Warm Up" label
    u8g.setPrintPos(45, 50);
    u8g.print("Warm Up");
  }

  RTC.readClock();//Read the time from real time clock and print it on Screen
  u8g.setFont(u8g_font_04b_03);
  u8g.setPrintPos(56, 58);
  u8g.print(String(RTC.getHours()) + ":" + String(RTC.getMinutes()));
  u8g.setPrintPos(49, 64);
  u8g.print(String(RTC.getDate()) + "/" + String(RTC.getMonth()) + "/" +
String(RTC.getYear()));
}




void insertuser() {//Insert User function
  Key = Kpd.getKey();//Read the button of the keypad
  if (Key) {
    if (Key >= '0' && Key <= '9' && user.length() <= 4) {//If button of the keypad is number
add it on user variable
      user = user + Key;
    }
    if (Key == 'C')user = "";//if button ic "C" clear the user variable
    if (Key == 'B') {//if button ic "B" clear the user variable and go to Main Screen
      user = "";
      screen = 0;
    }
    if (Key == 'A') {//if button ic "A" and if the user lenght is 5 digit storage the username
and go to Main Screen
      if (user.length() == 5) {
        screen = 0;
      }
    }
  }
}
```

```
void insertuserscreen() {
  u8g.setFont(u8g_font_6x10);//Set the Font of the Screen

  u8g.setPrintPos(19, 7);//Print labels and username
  u8g.print("INSERT USERNAME");

  u8g.setPrintPos(0, 36);
  u8g.print("USERNAME:");
  u8g.setPrintPos(55, 36);
  u8g.print(user);

  u8g.setPrintPos(10, 62);
  u8g.print("Must be five digit");
}




void datalogger() {//Data logger function to display the min and max of the sensors
  Key = Kpd.getKey();//Read the button of the keypad
  if (Key) {
    if (Key == 'B')screen = 0;//if button is "B" go to Main Screen
  }
}

void dataloggerscreen() {
  u8g.setFont(u8g_font_6x10);//Set the Font of the Screen

  u8g.setPrintPos(28, 7);//Print "DATA LOGGING" label
  u8g.print("DATA LOGGING");

  u8g.setPrintPos(0, 18);//Print max,min of the potentiometer
  u8g.print("Min:" + String(pot4minvalue));
  u8g.setPrintPos(0, 28);
  u8g.print("Max:" + String(pot4maxvalue));

  u8g.setPrintPos(0, 53);//Print max,min of the potentiometer
  u8g.print("Min:" + String(pot3minvalue));
```

```
  u8g.setPrintPos(0, 63);
  u8g.print("Max:" + String(pot3maxvalue));

  if (pot2minvalue >= 100)u8g.setPrintPos(86, 18);//Print max,min of the potentiometer
  else if (pot2minvalue >= 10)u8g.setPrintPos(92, 18);
  else if (pot2minvalue >= 0)u8g.setPrintPos(98, 18);
  u8g.print("Min:" + String(pot2minvalue));
  if (pot2maxvalue >= 100)u8g.setPrintPos(86, 28);
  else if (pot2maxvalue >= 10)u8g.setPrintPos(92, 28);
  else if (pot2maxvalue >= 0)u8g.setPrintPos(98, 28);
  u8g.print("Max:" + String(pot2maxvalue));

  if (pot1minvalue >= 100)u8g.setPrintPos(86, 53);//Print max,min of the potentiometer
  else if (pot1minvalue >= 10)u8g.setPrintPos(92, 53);
  else if (pot1minvalue >= 0)u8g.setPrintPos(98, 53);
  u8g.print("Min:" + String(pot1minvalue));
  if (pot1maxvalue >= 100)u8g.setPrintPos(86, 63);
  else if (pot1maxvalue >= 10)u8g.setPrintPos(92, 63);
  else if (pot1maxvalue >= 0)u8g.setPrintPos(98, 63);
  u8g.print("Max:" + String(pot1maxvalue));

  u8g.setPrintPos(50, 29);//Print "Score" label
  u8g.print("Score");

  u8g.setFont(u8g_font_helvB12);//Set the Font of the Screen

  if (score >= 1000)u8g.setPrintPos(46, 43);//Print the Score of the user
  else if (score >= 100)u8g.setPrintPos(51, 43);
  else if (score >= 10)u8g.setPrintPos(55, 43);
  else if (score >= 0)u8g.setPrintPos(60, 43);
  u8g.print(String(score));
}




void settings() {//Settings Function
  Key = Kpd.getKey();//Read the button of the keypad
  if (Key) {
```

```
  if (Key == 'B') {//if button is "B" go to Main Screen
    settingsmenu = 0;
    screen = 0;
  }
  if (Key == '2')settingsmenu--;//if button is "2" move up the cursor
  if (Key == '8')settingsmenu++;//if button is "8" move down the cursor
  if (Key == 'A') {//if Button is "A" select the item wich is after the cursor

    switch (settingsmenu) {
      case 0://if it's select the warm-up time, change the warm-up time and storage the
value on eeprom
        warmuptime++;
        if (warmuptime > 5)warmuptime = 1;
        EEPROM.write(0, warmuptime);
        break;
      case 1://if it's select the BuzzerEnable,change the BuzzerEnable and storage the
value on eeprom
        BuzzerEnable++;
        if (BuzzerEnable > 1)BuzzerEnable = 0;
        EEPROM.write(1, BuzzerEnable);
        break;
      case 2://if it's select the SdWriteEnable,change the SdWriteEnable and storage the
value on eeprom
        SdWriteEnable++;
        if (SdWriteEnable > 1)SdWriteEnable = 0;
        EEPROM.write(2, SdWriteEnable);
        break;
      case 3://If it's select Time Setup go to RtcSetup screen to set the time
        RTC.readClock();//Read the time from real time clock
        datesetup = String(RTC.getDate());
        monthsetup = String(RTC.getMonth());
        yearsetup = String(RTC.getYear());
        hoursetup = String(RTC.getHours());
        minutessetup = String(RTC.getMinutes());
        screen = 5;
        break;
    }

  }
 }
```

```
  if (settingsmenu < 0)settingsmenu = 0;
  if (settingsmenu > 3)settingsmenu = 3;
}

void settingsscreen() {
  u8g.setFont(u8g_font_6x10);//Set the Font of the Screen

  u8g.setPrintPos(40, 7);//Print "SETTINGS" label
  u8g.print("SETTINGS");

  u8g.setPrintPos(10, 20);//Print WarmUp Minutes variable
  u8g.print("WarmUp Minutes:");
  u8g.setPrintPos(110, 20);
  u8g.print(String(warmuptime));

  u8g.setPrintPos(10, 32);//Print Buzzer Enable variable
  u8g.print("Buzzer Enable:");
  u8g.setPrintPos(110, 32);
  u8g.print(String(BuzzerEnable));

  u8g.setPrintPos(10, 44);//Print Sd Enable variable
  u8g.print("Sd Enable:");
  u8g.setPrintPos(110, 44);
  u8g.print(String(SdWriteEnable));

  u8g.setPrintPos(10, 56);//Print "Time Setup>>" label
  u8g.print("Time Setup>>");

  switch (settingsmenu) {//Print the cursor
    case 0:
      u8g.setPrintPos(0, 20);
      break;
    case 1:
      u8g.setPrintPos(0, 32);
      break;
    case 2:
      u8g.setPrintPos(0, 44);
      break;
    case 3:
      u8g.setPrintPos(0, 56);
```

```
    break;
  }
  u8g.print(">");
}




void RtcSetup() {//Function to Set the time on real time clock
  Key = Kpd.getKey();
  if (Key) {
    if (Key == 'B') {//if button is "B" go to Main Screen
      if (datesetup.toInt() <= 31 && datesetup.toInt() >= 1 && monthsetup.toInt() <= 12 &&
monthsetup.toInt() >= 1 && yearsetup.toInt() >= 16 && yearsetup.toInt() <= 99 &&
hoursetup.toInt() <= 24 && minutessetup.toInt() <= 59) {
        RTC.start();
        RTC.switchTo24h();
        switch (Rtcscreen) {
          case 0:
            RTC.setDate(datesetup.toInt());
            break;
          case 1:
            RTC.setMonth(monthsetup.toInt());
            break;
          case 2:
            RTC.setYear(yearsetup.toInt());
            break;
          case 3:
            RTC.setHours(hoursetup.toInt());
            break;
          case 4:
            RTC.setMinutes(minutessetup.toInt());
            RTC.setSeconds(0);
            break;
        }
        RTC.setClock();
        RTC.readClock();
      }
      datesetup = String(RTC.getDate());
```

```
    monthsetup = String(RTC.getMonth());
    yearsetup = String(RTC.getYear());
    hoursetup = String(RTC.getHours());
    minutessetup = String(RTC.getMinutes());
    Rtcscreen--;
    if (Rtcscreen < 0) {
     Rtcscreen = 0;
     screen = 3;
    }
   }
   if (Key == 'A') {//if button is "A" go to the next variable
    if (datesetup.toInt() <= 31 && datesetup.toInt() >= 1 && monthsetup.toInt() <= 12 &&
monthsetup.toInt() >= 1 && yearsetup.toInt() >= 16 && yearsetup.toInt() <= 99 &&
hoursetup.toInt() <= 24 && minutessetup.toInt() <= 59) {
     RTC.start();
     RTC.switchTo24h();
     switch (Rtcscreen) {
      case 0:
       RTC.setDate(datesetup.toInt());
       break;
      case 1:
       RTC.setMonth(monthsetup.toInt());
       break;
      case 2:
       RTC.setYear(yearsetup.toInt());
       break;
      case 3:
       RTC.setHours(hoursetup.toInt());
       break;
      case 4:
       RTC.setMinutes(minutessetup.toInt());
       RTC.setSeconds(0);
       break;
     }
     RTC.setClock();
     RTC.readClock();
    }
    datesetup = String(RTC.getDate());
    monthsetup = String(RTC.getMonth());
    yearsetup = String(RTC.getYear());
```

```
    hoursetup = String(RTC.getHours());
    minutessetup = String(RTC.getMinutes());
    Rtcscreen++;
    if (Rtcscreen > 4) {
      Rtcscreen = 0;
      screen = 3;
    }
  }
  if (Key == 'C') {//if button is "C" clear the variable
    switch (Rtcscreen) {
      case 0:
        datesetup = "";
        break;
      case 1:
        monthsetup = "";
        break;
      case 2:
        yearsetup = "";
        break;
      case 3:
        hoursetup = "";
        break;
      case 4:
        minutessetup = "";
        break;
    }
  }
  if (Key >= '0' && Key <= '9') {//if button is number set the value
    switch (Rtcscreen) {
      case 0:
        if (datesetup.length() <= 1)datesetup = datesetup + Key;
        break;
      case 1:
        if (monthsetup.length() <= 1)monthsetup = monthsetup + Key;
        break;
      case 2:
        if (yearsetup.length() <= 1)yearsetup = yearsetup + Key;
        break;
      case 3:
        if (hoursetup.length() <= 1)hoursetup = hoursetup + Key;
```

```
        break;
       case 4:
        if (minutessetup.length() <= 1)minutessetup = minutessetup + Key;
        break;
     }
   }
  }
}

void RtcSetupScreen() {
  u8g.setFont(u8g_font_6x10);//Set the Font of the Screen

  u8g.setPrintPos(34, 7);//Print "TIME SETUP" label
  u8g.print("TIME SETUP");

  u8g.setPrintPos(0, 38);//Print variable on different screens
  switch (Rtcscreen) {
    case 0:
      u8g.print("Date:  " + datesetup);
      break;
    case 1:
      u8g.print("Month:  " + monthsetup);
      break;
    case 2:
      u8g.print("Year:  " + yearsetup);
      break;
    case 3:
      u8g.print("Hour:  " + hoursetup);
      break;
    case 4:
      u8g.print("Minutes:  " + minutessetup);
      break;
  }
}




void WriteToSd(byte Wu) {
```

```
  if (user.length() == 5 && userwithpath.length() <= 5) {//Check if the username is
correct and add the ending for the excel file
    userwithpath = user + ".CSV";
    userwithpath.toCharArray(filename, 10);
  }
  RTC.readClock();//Read the time from the real time clock
  if (filename != "") {
    if (! SD.exists(filename)) {//If the file name doesn't exists create the excel file and
write the label on the first line
      myFile = SD.open(filename, FILE_WRITE);
      if (myFile) {
        myFile.print("User");
        myFile.print(",");
        myFile.print("Score");
        myFile.print(",");
        myFile.print("Date");
        myFile.print(",");
        myFile.print("Time");
        myFile.print(",");
        myFile.print("RiSh");
        myFile.print(",");
        myFile.print("RiShMi");
        myFile.print(",");
        myFile.print("RiShMa");
        myFile.print(",");
        myFile.print("RiEl");
        myFile.print(",");
        myFile.print("RiElMi");
        myFile.print(",");
        myFile.print("RiElMa");
        myFile.print(",");
        myFile.print("LeSh");
        myFile.print(",");
        myFile.print("LeShMi");
        myFile.print(",");
        myFile.print("LeShMa");
        myFile.print(",");
        myFile.print("LeEl");
        myFile.print(",");
        myFile.print("LeElMi");
```

```
    myFile.print(",");
    myFile.print("LeElMa");
    myFile.println();
    myFile.close();
  } else screen = 4;
}
myFile = SD.open(filename, FILE_WRITE);
if (myFile) {//If the file is open write the data on the excel file
  myFile.print(user);
  myFile.print(",");
  myFile.print(score);
  myFile.print(",");
  myFile.print(RTC.getDate());
  myFile.print("/");
  myFile.print(RTC.getMonth());
  myFile.print("/");
  myFile.print(RTC.getYear());
  myFile.print(",");
  myFile.print(RTC.getHours());
  myFile.print(":");
  myFile.print(RTC.getMinutes());
  myFile.print(":");
  myFile.print(RTC.getSeconds());
  myFile.print(",");
  myFile.print(pot1value);
  myFile.print(",");
  myFile.print(pot1minvalue);
  myFile.print(",");
  myFile.print(pot1maxvalue);
  myFile.print(",");
  myFile.print(pot2value);
  myFile.print(",");
  myFile.print(pot2minvalue);
  myFile.print(",");
  myFile.print(pot2maxvalue);
  myFile.print(",");
  myFile.print(pot3value);
  myFile.print(",");
  myFile.print(pot3minvalue);
  myFile.print(",");
```

```
      myFile.print(pot3maxvalue);
      myFile.print(",");
      myFile.print(pot4value);
      myFile.print(",");
      myFile.print(pot4minvalue);
      myFile.print(",");
      myFile.print(pot4maxvalue);
      if (Wu == 1)myFile.print(",Warm Up");
      myFile.println();
      myFile.close();
    } else screen = 4;
  }
}




void sderror() {//Sd error function
  if (SD.begin(chipSelectpin))screen = 0;//If SD error is fixed go to Main Screen
}

void sderrorscreen() {
  u8g.setFont(u8g_font_6x10);//Set the Font of the Screen

  u8g.setPrintPos(40, 36);//Print the "Sd Error" label
  u8g.print("Sd Error");
}
```

9. ΠΑΡΑΡΤΗΜΑ Β΄

ΠΗΓΑΙΟΣ ΚΩΔΙΚΑΣ ΠΕΡΙΒΛΗΜΑΤΟΣ ΗΛΕΚΤΡΩΝΙΚΩΝ ΕΞΑΡΤΗΜΑΤΩΝ

```
// [Box dimensions]
// Length
Length       = 50;
// Width
Width        = 180;
// Height
Height       = 100;
// Wall thickness
Thick        = 2;//[2:5]

// [Box options]
//Filet diameter
Filet        = 2;//[0.1:12]
//Filet smoothness
Resolution   = 50;//[1:100]
//Tolerance (Panel/rails gap)
m            = 0.9;
//Pieds PCB - PCB feet (x4)
PCBFeet      = 0;// [0:No, 1:Yes]
//Decorations to ventilation holes
Vent         = 1;// [0:No, 1:Yes]
//Holes width (in mm)
Vent_width   = 1.5;



//Low left corner X position
PCBPosX      = 7;
//Low left corner Y position
PCBPosY      = 6;
//PCB Length
PCBLength    = 70;
//PCB Width
PCBWidth     = 50;
```

```
//Feet height
FootHeight      = 10;
//Foot diameter
FootDia         = 8;
//Hole diameter
FootHole        = 3;



// [STL element to export]
//Top shell
TShell          = 0;// [0:No, 1:Yes]
//Bottom shell
BShell          = 0;// [0:No, 1:Yes]
//Front panel
FPanL           = 1;// [0:No, 1:Yes]
//Back panel
BPanL           = 0;// [0:No, 1:Yes]




// [Hidden]
//Shell color
Couleur1        = "Orange";
//Panels color
Couleur2        = "OrangeRed";
//Making decorations thicker if it is a vent to make sure they go through shell
Dec_Thick       = Vent ? Thick * 2 : Thick;
//Depth decoration
Dec_size        = Vent ? Thick * 2 : 0.8;







//////////Generic rounded box//////////
module RoundBox($a = Length, $b = Width, $c = Height) { // Cube bords arrondis
  $fn = Resolution;
  translate([0, Filet, Filet]) {
    minkowski () {
      cube ([$a - (Length / 2), $b - (2 * Filet), $c - (2 * Filet)], center = false);
```

```
    rotate([0, 90, 0]) {
      cylinder(r = Filet, h = Length / 2, center = false);
    }
   }
  }
}


////////////////////////////////Module Shell////////////////////////////////////
module Coque() {
 Thick = Thick * 2;
 difference() {
  difference() { //sides decoration
   union() {
    difference() {//Substraction Fileted box
     difference() { //Median cube slicer
      union() {
       difference() {
        RoundBox();
        translate([Thick / 2, Thick / 2, Thick / 2]) {
         RoundBox($a = Length - Thick, $b = Width - Thick, $c = Height - Thick);
        }
       }
       difference() { //largeur Rails
        translate([Thick + m, Thick / 2, Thick / 2]) { // Rails
         RoundBox($a = Length - ((2 * Thick) + (2 * m)), $b = Width - Thick, $c =
Height - (Thick * 2));
        }//fin Rails
        translate([((Thick + m / 2) * 1.55), Thick / 2, Thick / 2 + 0.1]) { // +0.1 added to
avoid the artefact
         RoundBox($a = Length - ((Thick * 3) + 2 * m), $b = Width - Thick, $c = Height
- Thick);
        }
       }
      }
      translate([-Thick, -Thick, Height / 2]) {
       cube ([Length + 100, Width + 100, Height], center = false);
      }
     }//End Median cube slicer
     translate([-Thick / 2, Thick, Thick]) {
```

```
      RoundBox($a = Length + Thick, $b = Width - Thick * 2, $c = Height - Thick);
    }
  }


  difference() { // wall fixation box legs
    union() {
      translate([3 * Thick + 5, Thick, Height / 2]) {
        rotate([90, 0, 0]) {
          $fn = 6;
          cylinder(d = 16, Thick / 2);
        }
      }

      translate([Length - ((3 * Thick) + 5), Thick, Height / 2]) {
        rotate([90, 0, 0]) {
          $fn = 6;
          cylinder(d = 16, Thick / 2);
        }
      }

    }
    translate([4, Thick + Filet, Height / 2 - 57]) {
      rotate([45, 0, 0]) {
        cube([Length, 40, 40]);
      }
    }
    translate([0, -(Thick * 1.46), Height / 2]) {
      cube([Length, Thick * 2, 10]);
    }
  } //Fin fixation box legs
}

union() { // outbox sides decorations

  for (i = [0:Thick:Length / 4]) {


    translate([10 + i, -Dec_Thick + Dec_size, 1]) {
      cube([Vent_width, Dec_Thick, Height / 4]);
```

```
        }
        translate([(Length - 10) - i, -Dec_Thick + Dec_size, 1]) {
          cube([Vent_width, Dec_Thick, Height / 4]);
        }
        translate([(Length - 10) - i, Width - Dec_size, 1]) {
          cube([Vent_width, Dec_Thick, Height / 4]);
        }
        translate([10 + i, Width - Dec_size, 1]) {
          cube([Vent_width, Dec_Thick, Height / 4]);
        }
      }
    }
  }


  union() { //sides holes
    $fn = 50;
    translate([3 * Thick + 5, 20, Height / 2 + 4]) {
      rotate([90, 0, 0]) {
        cylinder(d = 2, 20);
      }
    }
    translate([Length - ((3 * Thick) + 5), 20, Height / 2 + 4]) {
      rotate([90, 0, 0]) {
        cylinder(d = 2, 20);
      }
    }
    translate([3 * Thick + 5, Width + 5, Height / 2 - 4]) {
      rotate([90, 0, 0]) {
        cylinder(d = 2, 20);
      }
    }
    translate([Length - ((3 * Thick) + 5), Width + 5, Height / 2 - 4]) {
      rotate([90, 0, 0]) {
        cylinder(d = 2, 20);
      }
    }
  }
 }
}
```

```
////////////////////////// - Foot with base filet - ////////////////////////////////
module foot(FootDia, FootHole, FootHeight) {
  Filet = 2;
  color(Couleur1)
  translate([0, 0, Filet - 1.5])
  difference() {
    difference() {
      cylinder(d = FootDia + Filet, FootHeight - Thick, $fn = 100);
      rotate_extrude($fn = 100) {
        translate([(FootDia + Filet * 2) / 2, Filet, 0]) {
          minkowski() {
            square(10);
            circle(Filet, $fn = 100);
          }
        }
      }
    }
    cylinder(d = FootHole, FootHeight + 1, $fn = 100);
  }
}

module Feet() {
  ////////////////////// - PCB only visible in the preview mode - ///////////////////////
  translate([3 * Thick + 2, Thick + 5, FootHeight + (Thick / 2) - 0.5]) {
    % square ([PCBLength + 10, PCBWidth + 10]);
    translate([PCBLength / 2, PCBWidth / 2, 0.5]) {
      color("Olive")
      % text("PCB", halign = "center", valign = "center", font = "Arial black");
    }
  }


  ////////////////////////////// - 4 Feet - ////////////////////////////////////////
  translate([3 * Thick + 7, Thick + 10, Thick / 2]) {
    foot(FootDia, FootHole, FootHeight);
  }
  translate([(3 * Thick) + PCBLength + 7, Thick + 10, Thick / 2]) {
    foot(FootDia, FootHole, FootHeight);
  }
  translate([(3 * Thick) + PCBLength + 7, (Thick) + PCBWidth + 10, Thick / 2]) {
```

```
    foot(FootDia, FootHole, FootHeight);
  }
  translate([3 * Thick + 7, (Thick) + PCBWidth + 10, Thick / 2]) {
    foot(FootDia, FootHole, FootHeight);
  }

}




////////////////////////////////////////////////////////////////////
////////////////////// <- Holes Panel Manager -> //////////////////////
////////////////////////////////////////////////////////////////////
//                      <- Panel ->
module Panel(Length, Width, Thick, Filet) {
  scale([0.5, 1, 1])
  minkowski() {
    cube([Thick, Width - (Thick * 2 + Filet * 2 + m), Height - (Thick * 2 + Filet * 2 + m)]);
    translate([0, Filet, Filet])
    rotate([0, 90, 0])
    cylinder(r = Filet, h = Thick, $fn = 100);

  }
}




//                      <- Circle hole ->
// Cx=Cylinder X position | Cy=Cylinder Y position | Cdia= Cylinder dia | Cheight=Cyl
height
module CylinderHole(OnOff, Cx, Cy, Cdia) {
  if (OnOff == 1)
    translate([Cx, Cy, -1])
    cylinder(d = Cdia, 10, $fn = 50);
}
//                      <- Square hole ->
// Sx=Square X position | Sy=Square Y position | Sl= Square Length | Sw=Square Width
| Filet = Round corner
module SquareHole(OnOff, Sx, Sy, Sl, Sw, Filet) {
```

```
  if (OnOff == 1)
    minkowski() {
    translate([Sx + Filet / 2, Sy + Filet / 2, -1])
    cube([Sl - Filet, Sw - Filet, 10]);
    cylinder(d = Filet, h = 10, $fn = 100);
  }
}



//                  <- Linear text panel ->
module LText(OnOff, Tx, Ty, Font, Size, Content) {
  if (OnOff == 1)
    translate([Tx, Ty, Thick + .5])
    linear_extrude(height = 0.5) {
    text(Content, size = Size, font = Font);
  }
}
//                  <- Circular text panel->
module CText(OnOff, Tx, Ty, Font, Size, TxtRadius, Angl, Turn, Content) {
  if (OnOff == 1) {
    Angle = -Angl / len(Content);
    translate([Tx, Ty, Thick + .5])
    for (i = [0:len(Content) - 1] ) {
      rotate([0, 0, i * Angle + 90 + Turn])
      translate([0, TxtRadius, 0]) {
        linear_extrude(height = 0.5) {
          text(Content[i], font = Font, size = Size,  valign = "baseline", halign = "center");
        }
      }
    }
  }
}



////////////////////// <- New module Panel -> //////////////////////
module FPanL() {
  difference() {
```

```
   color(Couleur2)
   Panel(Length, Width, Thick, Filet);
   rotate([90, 0, 90]) {
    color(Couleur2) {
      //Start Back
      //CylinderHole(1,30,25,16);
      //CylinderHole(1,55,25,16);
      //CylinderHole(1,80,25,16);
      //CylinderHole(1,105,25,16);
      //CylinderHole(1,150,25,16);
      //CylinderHole(1,150,70,12.7);
      //End Back

      //Start Front
      SquareHole  (1, 15, 28, 70, 40, 4);    //(On/Off,Xpos,Ypos,Length,Width,Filet) auto
      SquareHole  (1, 116, 15, 35, 10, 3);
      //End Front
      //                    <- To here ->
    }
   }
  }

  color(Couleur1) {
   translate ([-.5, 0, 0])
   rotate([90, 0, 90]) {

    //Start Front
    LText(1, 4, 87, "Arial Black", 4, "VASILIS LASKARIDIS");
    LText(1, 4, 4, "Arial Black", 3, "AEI PEIRAIA T.T.  M.H.Y.S");
    //End Front
    //LText(1,120,83,"Arial Black",4,"KeyPad");
    //CText(1,93,29,"Arial Black",4,10,180,0,"1 . 2 . 3 . 4 . 5 . 6");//(On/Off, Xpos, Ypos,
"Font", Size, Diameter, Arc(Deg), Starting Angle(Deg),"Text")
   }
  }
}


////////////////////////// <- Main part -> //////////////////////
if (TShell == 1)
```

```
  //Top Shell
  color( Couleur1, 1) {
  translate([0, Width, Height + 0.2]) {
    rotate([0, 180, 180]) {
      Coque();
    }
  }
}

if (BShell == 1)
  //Bottom shell
  color(Couleur1) {
  Coque();
}

//PCB feet
if (PCBFeet == 1)
  translate([PCBPosX, PCBPosY, 0]) {
  Feet();
}

//Front panel
if (FPanL == 1)
  translate([Length-(Thick * 2 + m / 2), Thick + m / 2, Thick + m / 2])
  FPanL();

//Back panel
if (BPanL == 1)
  color(Couleur2)
  translate([Thick + m / 2, Thick + m / 2, Thick + m / 2])
  Panel(Length, Width, Thick, Filet);
```

10. ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] J. Zhang, G. G. Hanauer, “The Application of Mean Field Theory to Image Motion Estimation”, *IEEE Trans. on Image Processing*, vol. 4, no. 1, pp. 19-32, 1995.

[2] S. Z. Li, Markov Random Field Modeling in Computer Vision, Springer-Verlag, 1995.

[3] Ι. Μελετόπουλου, “Τεχνικές σύνταξης πτυχιακής εργασίας”, *Πτυχιακή εργασία*, Τμήμα Ηλεκτρονικών Υπολογιστών Συστημάτων, ΤΕΙ Πειραιά, 2004.

[4] Sims, N.R., Muyderman, & H. (2010). Mitochondria, oxidative metabolism and cell death in stroke. Biochimica et Biophysica Acta (BBA)-Molecular Basis of Disease, 1802(1), 80–91.

[5] T. Truelsen and R. Bonita, “The worldwide burden of stroke: current status and future projections,” in Handbook of Clinical Neurology Vol 92 (3rd Series): Stroke Part I: Basic and Epidemiological Aspects, M. Fisher, Ed. Amsterdam: Elsevier, 2009, pp. 327-336.

[6] K. Strong, C. Mathers, and R. Bonita, “Preventing stroke: saving lives around the world,” Lancet Neurology, vol. 6, no. 2, pp. 182-187, 2007.

[7] The Consensus Panel on the Stroke Rehabilitation System, Consensus Panel of the Stroke Rehabilitation System: ―Time is Function.‖Ottawa, ON, Canada: .

[8] P. Langhorne, F. Coupar, and A. Pollock, “Motor recovery after stroke: a systematic review,” Lancet Neurology, vol. 8, no. 8, pp. 741-754, 2009.

[9] http://www.papapostolou.gr/

[10] Σ. Ευστάθιος, “ Ανάπτυξη συστήματος μέτρησης του μονοξειδίου του άνθρακα και καταχώρηση των δεδομένων μέσω datalogger. ”, *Πτυχιακή εργασία*, Τμήμα Ηλεκτρονικών Υπολογιστών Συστημάτων, ΤΕΙ Πειραιά, 2016.

[11] https://deltahacker.gr/arduino-intro/

[12] https://el.wikipedia.org/wiki/Arduino