ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Συσκευή πραγματικού χρόνου αναγνώρισης της ελληνικής νοηματικής γλώσσας

Νικόλαος Χατζηιωάννου

ΕΠΙΒΛΕΠΟΝΤΕΣ ΚΑΘΗΓΗΤΕΣ
Αντώνιος Δαδαλιάρης
Γεώργιος Σταμούλης

ΙΟΥΛΙΟΣ 2017

Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις (1), που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί χωρίς να τα περικλείω σε εισαγωγικά και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.

2. Δέχομαι ότι η αυτολεξεί παράθεση χωρίς εισαγωγικά, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.

3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια

4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»

Περιεχόμενα

Εικόνες

Πίνακες

Λίστα Εξισώσεων

Ευχαριστίες

Η παρούσα πτυχιακή εργασία δεν θα μπορούσε να ολοκληρωθεί και να έχει αυτή τη μορφή χωρίς την υποστήριξη κάποιων ατόμων. Θα ήθελα να ευχαριστήσω θερμά τους καθηγητές μου, κ. Δαδαλιάρη Αντώνιο και κ. Σταμούλη Γεώργιο, για την ανάθεση του θέματος καθώς και για την πολύτιμη βοήθειά τους, την καθοδήγησή τους και τον χρόνο που αφιέρωσαν για τη διεκπεραίωση της εργασίας μου.

Επίσης, θα ήθελα να ευχαριστήσω του γονείς μου και τους φίλους μου, οι οποίοι με στήριξαν καθ' όλη τη διάρκεια των σπουδών μου με κάθε τρόπο.

1. Εισαγωγή

Η παρούσα πτυχιακή εργασία έχει ως αντικείμενο να δημιουργήσει μια συσκευή η οποία θα μπορεί να μεταφράζει τις χειρομορφές που εκτελούνται από νοηματιστές[1] στο νόημα που προσπαθούν να αποδώσουν. Πιο συγκεκριμένα, θα προσπαθήσουμε να αναγνωρίσουμε χειρομορφές που θα έχουν ως νόημά τους τα γράμματα του ελληνικού νοηματικού αλφάβητου. Η συσκευή αυτή έχει σκοπό να βοηθήσει τους ανθρώπους που ασκούν τη νοηματική γλώσσα προκειμένου να επικοινωνήσουν με άτομα που δεν γνωρίζουν τη νοηματική γλώσσα. Η πτυχιακή εργασία οργανώνεται στα εξής κεφάλαια: Εισαγωγή, Επιλογή σχεδιασμού, Μεθοδολογία, Αποτελέσματα, Συμπεράσματα και μελλοντικές επεκτάσεις. Η Εισαγωγή αναφέρεται στη νοηματική γλώσσα και αποσκοπεί στο να την κατανοήσουμε καλύτερα. Το δεύτερο κεφάλαιο, Επιλογή Σχεδιασμού, αναφέρεται στον τρόπο με τον οποίο καταλήξαμε στον τελικό σχεδιασμό της συσκευής, καθώς και στις απαιτήσεις που είχαμε εμείς, και θα μπορούσαν να έχουν και οι νοηματιστές, από μια τέτοια συσκευή. Στη Μεθοδολογία (Κεφάλαιο 3) παρουσιάζονται όλες οι απαραίτητες πληροφορίες σχετικά με τις συσκευές και τα λογισμικά που χρησιμοποιήθηκαν για την κατασκευή της συσκευής, καθώς επίσης και τα βήματα που ακολουθήθηκαν για να κατασκευαστεί. Το τέταρτο κεφάλαιο αναφέρεται στα αποτελέσματα της χρήσης της συσκευής και στο κατά πόσο εκπληρώθηκαν οι απαιτήσεις που είχαμε συζητήσει στο δεύτερο κεφάλαιο. Τέλος, στο τελευταίο κεφάλαιο αναφέρονται τα συμπεράσματα που προκύπτουν από τα προηγούμενα αποτελέσματα (Κεφάλαιο 4), όπως επίσης και οι επεκτάσεις που θα μπορούσε να έχει η συσκευή.

1.1 Νοηματική Γλώσσα

Είναι γνωστό ότι οι κωφοί και οι βαρήκοοι επικοινωνούν μεταξύ τους χρησιμοποιώντας μια γλώσσα διαφορετική από την καθομιλουμένη και συγκεκριμένα τη νοηματική. Η νοηματική γλώσσα αποτελείται από τις κινήσεις των χεριών, τη στάση και την κίνηση του σώματος και του προσώπου, καθώς και από

[1] Με τον όρο νοηματιστής εννοείται ο κωφός, ο βαρήκοος ή ο ακούοντας που χρησιμοποιεί τη νοηματική γλώσσα.

κάποια ηχητικά φωνήματα με σκοπό να φανεί η κίνηση των χειλιών για να αποδοθεί το σωστό νόημα και, κατ' επέκταση, να επιτευχθεί η επικοινωνία (Κέντρο Ελληνικής Νοηματικής Γλώσσας, 2017). Οι περισσότεροι άνθρωποι που δεν έχουν επαφές με κωφούς έχουν την τάση να πιστεύουν ότι όλες οι νοηματικές γλώσσες είναι ίδιες και ότι προσπαθούν να μιμηθούν τις εκάστοτε φωνούμενες γλώσσες, γεγονός το οποίο δεν ισχύει. Είναι σημαντικό να σημειωθεί ότι η νοηματική γλώσσα δεν είναι ίδια παντού, υπάρχουν διαφορές στη γλώσσα της κάθε χώρας αλλά και ιδιωματισμοί εντός της ίδιας χώρας, όπως ακριβώς συμβαίνει και με την καθομιλουμένη γλώσσα. Σε κάθε νοηματική γλώσσα που υπάρχει έχει γίνει εμφανές ότι κάποιες λέξεις που νοηματούνται είναι αποτέλεσμα μίμησης της καθημερινότητας χωρίς όμως αυτό να σημαίνει ότι ισχύει για το σύνολο των λέξεων της εκάστοτε νοηματικής γλώσσας (Καμαρινού & Μεϊμάρη, 2016). Τέλος, αξίζει να αναφερθεί ότι, όπως κάθε γλώσσα εξελίσσεται σύμφωνα με τις ανάγκες της κάθε εποχής, το ίδιο συμβαίνει και με τις νοηματικές γλώσσες.

1.2 Ελληνική Νοηματική Γλώσσα

Η Ελληνική Νοηματική Γλώσσα (Ε.Ν.Γ.) είναι η επίσημη, φυσική γλώσσα της ελληνικής κοινότητας των κωφών στην Ελλάδα. Η Ελληνική Νοηματική Γλώσσα, η οποία είναι μια πλήρης γλώσσα, διαφέρει τόσο από τις υπόλοιπες φωνούμενες γλώσσες όσο και από τις υπόλοιπες νοηματικές γλώσσες. Η νοηματική γλώσσα περιλαμβάνει λέξεις, προτάσεις και σύνταξη, ακριβώς όπως η καθομιλουμένη. Επίσης, όπως κάθε γλώσσα έχει τις ιδιαιτερότητες της, έτσι και αυτή. Για παράδειγμα, το ρήμα στη νοηματική γλώσσα πρέπει να μπαίνει πάντα στο τέλος, εκτός αν υπάρχει ερώτηση, οπότε σε αυτήν την περίπτωση τη θέση του ρήματος την παίρνει η ερωτηματική λέξη («πού πας;» → «πας πού;»). Επιπρόσθετα, πρέπει να γίνει κατανοητό ότι, όταν ένας κωφός στην καθημερινότητά του και στον προφορικό του λόγο χρησιμοποιεί αυτήν τη σύνταξη, του είναι δύσκολο να προσαρμοστεί στον γραπτό λόγο της εκάστοτε γλώσσας. Από τα παραπάνω φαίνεται το πόσο ολοκληρωμένες είναι οι νοηματικές γλώσσες.

Σε κάθε χώρα αναπτύσσεται η δική της νοηματική γλώσσα με διαφορετικά αλφάβητα και νοήματα. Αρκετές φορές υπάρχουν κοινά χαρακτηριστικά, αλλά

υπάρχουν συνήθως σημαντικές μορφολογικές διαφορές μεταξύ των νοηματικών γλωσσών, που εξαρτώνται από την ιδιομορφία της χώρας αλλά και από την ίδια τη γλώσσα, κάτι το οποίο παρατηρείται και στους ιδιωματισμούς της κάθε περιοχής της εκάστοτε χώρας. Για παράδειγμα, τη λέξη «μαύρο» η Αθήνα τη νοηματίζει διαφορετικά από ότι η Θεσσαλονίκη. Επομένως, ένας γνώστης της Ε.Ν.Γ. μπορεί κάποια νοήματα να μην τα αντιλαμβάνεται γιατί είναι ιδιωματισμοί κάποιας περιοχής και για αυτόν τον λόγο του είναι ακόμα πιο δύσκολο να κατανοήσει έναν άλλον γνώστη που νοηματεί στην αγγλική νοηματική γλώσσα. Οπότε, η Ε.Ν.Γ., όπως και όλες οι υπόλοιπες νοηματικές γλώσσες, έχει ένα διαφορετικό γλωσσικό σύστημα με διαφορετικούς γλωσσικούς κανόνες, σύνταξη και ρίζα, το οποίο δεν βασίζεται στην ελληνική γλώσσα. Παρ' όλα αυτά, γίνεται προσπάθεια να δημιουργηθεί μια διεθνής νοηματική γλώσσα η οποία θα χρησιμοποιεί κοινές λέξεις για όλες τις νοηματικές γλώσσες που υπάρχουν (π.χ. καρέκλα, τραπέζι, κ.λπ.). Με αυτόν τον τρόπο, θα αποτελεί έναν κώδικα φτιαγμένο ώστε να βοηθήσει στην απλή καθημερινή επικοινωνία μεταξύ ανθρώπων που νοηματούν σε διαφορετική γλώσσα.

Η επικοινωνία στις νοηματικές γλώσσες δεν επιτυγχάνεται μέσω του προφορικού λόγου, αλλά μέσω των κινήσεων και της όρασης, και πιο συγκεκριμένα, επιτυγχάνεται με τις χειρομορφές. Χειρομορφή είναι το σχήμα που παίρνει η παλάμη και η θέση που παίρνουν τα δάχτυλα. Βέβαια η ίδια χειρομορφή μπορεί να έχει πολλά διαφορετικά νοήματα και εξαρτάται από κάποιες παραμέτρους. Οι παράμετροι αυτές είναι οι εξής:

1. Ο προσανατολισμός της παλάμης, δηλαδή ανάλογα με την κατεύθυνση της παλάμης μπορούν να σχηματιστούν διαφορετικές λέξεις.
2. Η θέση της χειρομορφής σε σχέση με τον κορμό, το πρόσωπο ή τον χώρο, δηλαδή τα νοήματα δημιουργούνται εντός ενός προκαθορισμένου νοητού χώρου ο οποίος εκτείνεται από την κορυφή του κεφαλιού μέχρι τον κορμό του σώματος και περίπου 20 με 30 εκατοστά δεξιά και αριστερά από τους ώμους του νοηματιστή. Όλα αυτά, επίσης, σχηματίζουν διαφορετικές λέξεις σύμφωνα με τα παραπάνω.

3. Η κίνηση του χεριού, η οποία είναι απαραίτητη για να ολοκληρωθεί ένα νόημα. Επιπρόσθετα, η κίνηση μπορεί να δηλώνει τον αριθμό (ενικό ή πληθυντικό), το μέγεθος ενός αντικειμένου (μικρό ή μεγάλο, παχύ ή λεπτό), τη συχνότητα μιας ενέργειας, κάποια κίνηση («περπατώ», «ανεβαίνω στο τρένο», κ.λπ.), ακόμα και διαφορετικούς χρόνους (π.χ. Αόριστος).
4. Η στάση, η κίνηση του σώματος και η έκφραση του προσώπου (χείλη, ήχοι στόματος) αποτελούν απαραίτητα στοιχεία για την ολοκλήρωση του νοήματος, ώστε να δώσουν ένταση (συναισθήματα, ερωτήσεις, κ.ά.) στη γλώσσα, όπως γίνεται με τον τόνο της φωνής στις καθομιλουμένες γλώσσες. Για παράδειγμα, στην άρνηση του Αορίστου, εκτός από τη χειρομορφή και την κίνησή της, ο νοηματιστής πρέπει να προφέρει το φώνημα «απ» για να γίνει κατανοητό στον συνομιλητή του ότι αναφέρεται στο παρελθόν καθώς και ότι προσθέτει άρνηση στο νόημα.

Τέλος, το αλφάβητο, μπορεί να μην είναι κομμάτι της καθημερινότητας των κωφών, αλλά το χρησιμοποιούν για να αναφέρουν κάποιο όνομα εκτός του κύκλου τους[2] ή για να νοηματίσουν μια λέξη της οποίας το νόημα δεν γνωρίζουν (π.χ. αν δεν γνωρίζω το νόημα της λέξης «τζάκι» θα το νοηματίσω με τη χρήση του αλφάβητου).

1.3 Δυσκολίες της Ελληνικής Νοηματικής Γλώσσας

Εάν κάποιος θελήσει να μελετήσει μια νοηματική γλώσσα, θα δυσκολευτεί, διότι υπάρχει έλλειψη γραφής ή μεταγραφής κάποιου είδους. Αυτό είναι ένα πρόβλημα που εμφανίζεται σε πολλές νοηματικές και προφορικές γλώσσες, δηλαδή υπάρχουν ελάχιστες έως καθόλου καταγραφές της γλώσσας ή η μελέτη της είναι ιδιαίτερα περιορισμένη. Στην περίπτωση της νοηματικής γλώσσας, και συγκεκριμένα της Ε.Ν.Γ., με την οποία ασχολούμαστε, το πρόβλημα είναι ξεκάθαρο και το αντλούμε από το γεγονός ότι η καταγραφή οποιασδήποτε πληροφορίας γινόταν μέχρι τώρα μόνο με φωτογραφίες ή σκίτσα, από τα οποία έλειπε ένα βασικό συστατικό, η κίνηση. Παρ' όλα αυτά, κάποιες σχολές εκμάθησης της νοηματικής κάνουν

[2] Στον κύκλο των κωφών τα ονόματα είναι παρατσούκλια από διάφορα χαρακτηριστικά του καθενός, π.χ. σγουρά μαλλιά, μεγάλα μάτια, κ.ά.

προσπάθειες καταγραφής της γλώσσας με τη χρήση βίντεο που μπορεί να απεικονίσει τη χειρομορφή και όλες τις παραμέτρους που χρειάζονται (προσανατολισμός παλάμης, θέση χειρομορφής, κίνηση χεριού, στάση σώματος και εκφράσεις προσώπου) για να αποδοθεί το σωστό νόημα. Σε αυτό το σημείο αξίζει να αναφερθεί ότι έχουν γίνει κάποιες πρώιμες προσπάθειες για τη δημιουργία ηλεκτρονικού λεξικού μέσω android εφαρμογών (Μπιτσάκης, 2015). Επιπρόσθετα, διάφορες γλωσσικές και κοινωνικές προκαταλήψεις, όπως ότι η νοηματική δεν είναι «γλώσσα» ή ότι προσπαθεί να μιμηθεί την καθημερινότητα των ανθρώπων, έχουν εμποδίσει την ευρύτερη διάδοσή της. Δυστυχώς, ακόμα και σήμερα, ενώ υπάρχει ενδιαφέρον για την εκμάθηση της νοηματικής γλώσσας, υπάρχει έλλειψη βιβλίου γραμματικής ή εγχειριδίου εκμάθησης της γλώσσας, με αποτέλεσμα η γλώσσα να διδάσκεται με προφορικό τρόπο και η κατάκτησή της να είναι ακόμα πιο δύσκολη.

Εικόνα 1. Το ελληνικό νοηματικό αλφάβητο [3]

[3] Η εικόνα προέρχεται από Κέντρο Ελληνικής Νοηματικής γλώσσας, 2017

2. Επιλογή Σχεδιασμού

Σε αυτό το κεφάλαιο θα αναφερθούμε στον σχεδιασμό της συσκευής και στις παραμέτρους που μας βοήθησαν να καταλήξουμε σε αυτόν. Όπως θα φανεί και στη συνέχεια, ο σχεδιασμός του γαντιού επηρεάστηκε από διάφορους παράγοντες, (π.χ. τεχνικούς), από τα μειονεκτήματα ή τα πλεονεκτήματα που παρουσίασαν παρόμοιες συσκευές αναγνώρισης σε ξένες γλώσσες, αλλά και από τις απαιτήσεις που θα είχε ένας νοηματιστής από τη συσκευή.

2.1 Απαιτήσεις Νοηματιστών

Μία από τις απαιτήσεις που θα μπορούσε να έχει ο νοηματιστής είναι η αξιοπιστία στη μετάφραση ενός γράμματος, μιας λέξης ή μιας φράσης. Αυτό περιλαμβάνει το κατά πόσο θα είναι σωστές οι προβλέψεις που θα κάνει η συσκευή μας και το πόσο «ακριβής» θα είναι η μετάφραση που θα κάνει η συσκευή σε σχέση με το πραγματικό νόημα που θέλει να «πει» ο νοηματιστής. Επίσης, η ταχύτητα της μετάφρασης είναι ένας ακόμη πολύ σημαντικός παράγοντας, καθώς ένας νοηματιστής εκτελεί αρκετές κινήσεις το δευτερόλεπτο και η συσκευή θα πρέπει να είναι σε θέση να τις μεταφράσει. Αυτές οι δύο απαιτήσεις αποτελούν τον δείκτη αξιοπιστίας της συσκευής μας, παρ' όλα αυτά εμείς στη συγκεκριμένη συσκευή θα προσπαθήσουμε να αναγνωρίσουμε μόνο γράμματα του αλφάβητου, οπότε η ταχύτητα δεν θα αποτελεί απαραίτητη προϋπόθεση.

2.2 Άλλες απαιτήσεις

Εκτός από τις απαιτήσεις που θα μπορούσε να έχει ένας νοηματιστής, και οι οποίες αφορούν κυρίως τη μετάφραση της νοηματικής γλώσσας, υπάρχουν και άλλοι παράγοντες που θα πρέπει να έχουμε υπ' όψιν μας, όπως:

- **Οικονομία**: Το τελικό προϊόν θα πρέπει να είναι προσιτό στο καταναλωτικό κοινό.
- **Ευκολία στη χρήση**: Η συσκευή και το πρόγραμμα δεν θα πρέπει να έχει ιδιαίτερες απαιτήσεις από τον τελικό χρήστη. Ο χρήστης θα πρέπει να είναι

σε θέση να ενεργοποιεί και να χρησιμοποιεί το γάντι στις καθημερινές του ασχολίες χωρίς καθυστερήσεις ή τυχόν προβλήματα. Επίσης, το γάντι θα πρέπει να φοριέται εύκολα και να ξεκινάει η μετάφραση αμέσως μόλις φορεθεί.

- **Αξιοπιστία και ταχύτητα:** Η συσκευή θα πρέπει να είναι αξιόπιστη, δηλαδή να υπάρχουν ελάχιστα λάθη στη μετάφραση. Επίσης, θα πρέπει να είναι γρήγορη και να μεταφράζει ταυτόχρονα με τις κινήσεις του χρήστη.
- **Αισθητική:** Το γάντι, για να μπορέσει να επιτύχει ως προϊόν, θα πρέπει να είναι αισθητικά ευχάριστο αλλά και να μην προκαλεί καμία δυσφορία στον χρήστη που το χρησιμοποιεί.

2.3 Τελικός Σχεδιασμός

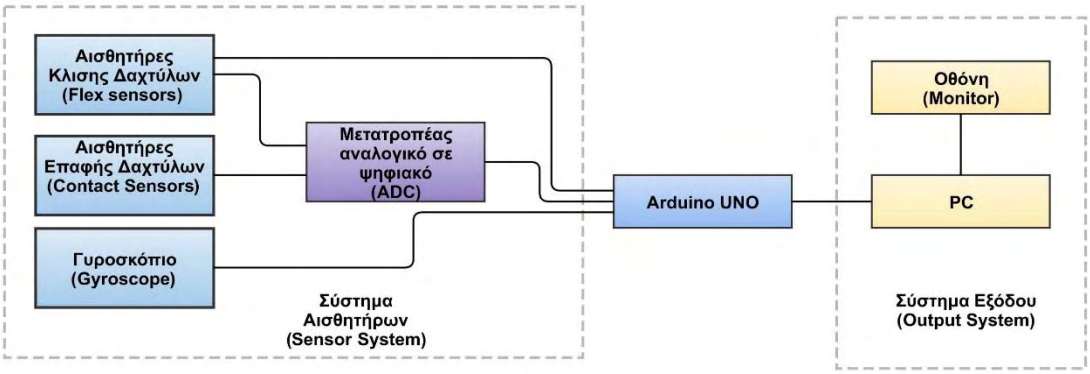
Ο σχεδιασμός που αποφασίστηκε περιλαμβάνει το γάντι, πάνω στο οποίο βρίσκονται όλοι οι απαραίτητοι αισθητήρες, ενώ όλοι οι απαραίτητοι υπολογισμοί θα γίνουν με έναν μικροελεγκτή (microcontroller) πάνω του. Το γάντι είναι συνδεδεμένο μέσω καλωδίου στον υπολογιστή από τον οποίο τροφοδοτείται με ρεύμα. Μόλις επιτευχθεί η επικοινωνία, ο υπολογιστής, μέσω της οθόνης, θα υποδείξει το αποτέλεσμα της μετάφρασης που έχει ήδη υπολογιστεί από τον μικροελεγκτή (microcontroller).

Ο σχεδιασμός που αποφασίσαμε δεν περιλαμβάνει όλες τις απαιτήσεις που αναφέραμε πιο πάνω, αλλά η συγκεκριμένη συσκευή αποτελεί έναν πρωτότυπο (prototype) σχεδιασμό που έχει στόχο του να ελέγξει το κατά πόσο μπορεί να αποτελέσει τη βάση για τη δημιουργία μιας πιο τελειοποιημένης συσκευής. Η νέα αυτή συσκευή θα μπορεί να συνοδεύσει έναν νοηματιστή σε όλες τις καθημερινές του ασχολίες.

Μετά από έρευνα προηγούμενων σχεδίων που αναγνώριζαν γράμματα σε ξένες νοηματικές γλώσσες και μέσα από τις συζητήσεις μου με τους επιβλέποντες καθηγητές μου, αποφασίσαμε ότι ο σχεδιασμός για την αναγνώριση του ελληνικού νοηματικού αλφάβητου θα πρέπει να περιέχει:

1. Αισθητήρες που θα μετρούν την κλίση των δαχτύλων

2. Αισθητήρες επαφής μεταξύ των δαχτύλων
3. Κατεύθυνση και κλίση της παλάμης



Εικόνα 2. Διάγραμμα σχεδίου συσκευής

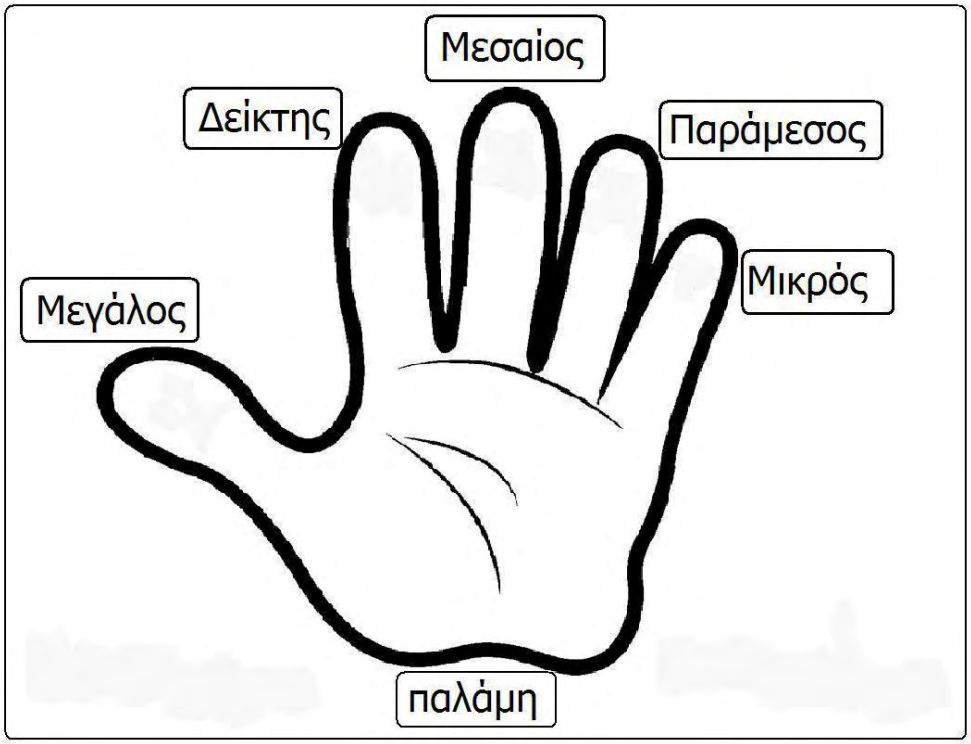
Η τελική συσκευή αποτελείται από το γάντι, τους αισθητήρες flex που μετράνε την κλίση των δαχτύλων, τους αισθητήρες επαφής μεταξύ των δαχτύλων, το γυροσκόπιο για την κλίση και την κατεύθυνση της παλάμης, καθώς και την πλακέτα (protoboard) που έχει τον μικροελεγκτή (microcontroller) και όλα τα αναγκαία εξαρτήματα.

2.3.1 Σύστημα Αισθητήρων

Το σύστημα αισθητήρων αποτελείται από τρεις βασικές κατηγορίες αισθητήρων. Η κάθε κατηγορία είναι απαραίτητη για τη λειτουργία της συσκευής καθώς επιστρέφει τα δικά της δεδομένα που είναι απαραίτητα για την αναγνώριση του αλφάβητου.

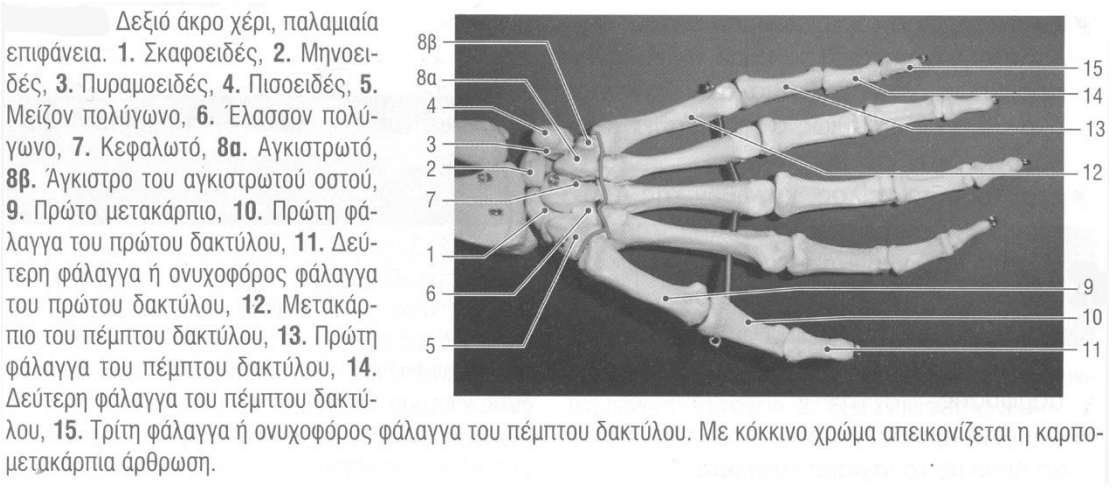
Η πρώτη κατηγορία είναι οι αισθητήρες που μετρούν την κλίση των δαχτύλων (flex sensors). Όπως αναφέραμε και στο πρώτο κεφάλαιο, η επικοινωνία μεταξύ των νοηματιστών βασίζεται σε έναν μεγάλο βαθμό στη διάταξη των δαχτύλων του νοηματιστή. Βάση της κλίσης των δαχτύλων του νοηματιστή σχηματίζεται και διαφορετική λέξη ή, στη δικιά μας περίπτωση, διαφορετικό γράμμα. Στην αρχή είχαμε αποφασίσει το κάθε δάχτυλο να έχει έναν δικό του αισθητήρα κλίσης (flex sensor). Το «μεγάλο» και το «μικρό» δάχτυλο θα είχαν

αισθητήρες μεγέθους 2.2 ιντσών ενώ τα υπόλοιπα δάχτυλα (δείκτης, μεσαίος, παράμεσος) από έναν αισθητήρα μεγέθους 4.5 ιντσών.



Εικόνα 3. Ονομασία δαχτύλων

Όμως, μετά την καλύτερη παρατήρηση του ελληνικού αλφάβητου των νοηματιστών, είδαμε ότι υπάρχουν περιπτώσεις κατά τις οποίες η κλίση των δαχτύλων είναι ιδιάζουσα και με έναν αισθητήρα δεν θα μπορούσαμε να έχουμε ακριβή αποτελέσματα. Για παράδειγμα, στο γράμμα «Δ» (Εικόνα 1), η πρώτη φάλαγγα του τρίτου δαχτύλου (μεσαίος) έχει μηδενική κλίση ενώ η δεύτερη και η τρίτη φάλαγγα έχουν κλίση. Με έναν αισθητήρα ανά δάχτυλο δεν θα ήμασταν σε θέση να ξεχωρίζαμε τέτοιες διαφορές. Οπότε αποφασίσαμε να αλλάξουμε τον αρχικό σχεδιασμό και να προσθέσουμε από έναν παραπάνω αισθητήρα 2.2 ιντσών στον δείκτη, στον μεσαίο και στον παράμεσο καθώς και να μην χρησιμοποιήσουμε αισθητήρες των 4.5 ιντσών αλλά των 2.2 ιντσών σε όλα τα δάχτυλα. Ο κάθε ένας από αυτούς τους αισθητήρες θα έχει το δικό του διαχωριστή τάσης [Διαχωριστής Τάσης (Voltage Divider)] και θα συνδέεται με τον μικροελεγκτή (microcontroller).



Δεξιό άκρο χέρι, παλαμιαία επιφάνεια. **1.** Σκαφοειδές, **2.** Μηνοειδές, **3.** Πυραμοειδές, **4.** Πισοειδές, **5.** Μείζον πολύγωνο, **6.** Έλασσον πολύγωνο, **7.** Κεφαλωτό, **8α.** Αγκιστρωτό, **8β.** Άγκιστρο του αγκιστρωτού οστού, **9.** Πρώτο μετακάρπιο, **10.** Πρώτη φάλαγγα του πρώτου δακτύλου, **11.** Δεύτερη φάλαγγα ή ονυχοφόρος φάλαγγα του πρώτου δακτύλου, **12.** Μετακάρπιο του πέμπτου δακτύλου, **13.** Πρώτη φάλαγγα του πέμπτου δακτύλου, **14.** Δεύτερη φάλαγγα του πέμπτου δακτύλου, **15.** Τρίτη φάλαγγα ή ονυχοφόρος φάλαγγα του πέμπτου δακτύλου. Με κόκκινο χρώμα απεικονίζεται η καρπομετακάρπια άρθρωση.

Εικόνα 4. Οστική παρουσίαση του δεξιού άνω άκρου[4]

Η δεύτερη κατηγορία αισθητήρων περιλαμβάνει τους αισθητήρες επαφής μεταξύ των δαχτύλων (contact sensors). Όπως αναφέραμε και στο πρώτο κεφάλαιο, πολλές χειρονομίες, δηλαδή λέξεις, προτάσεις ή γράμματα, απαιτούν επαφή μεταξύ των δαχτύλων του χεριού. Για παράδειγμα, στο γράμμα «Ω» υπάρχει επαφή μεταξύ των δαχτύλων. Για αυτόν τον λόγο, αποφασίσαμε ότι, για να έχουμε όσο το δυνατόν καλύτερη μετάφραση των γραμμάτων, θα είναι απαραίτητο να γνωρίζουμε πότε υπάρχει επαφή μεταξύ των δαχτύλων. Στο χέρι τοποθετήθηκαν τέσσερις τέτοιοι αισθητήρες, ένας σε κάθε δάχτυλο, εκτός του αντίχειρα. Επιλέξαμε τους αισθητήρες FSR-402, καθώς, όπως και οι αισθητήρες κλίσης δαχτύλων, έτσι και αυτοί χρειάζονται ο καθένας τους από έναν διαχωριστή τάσης για συνδεθούν με τον μικροελεγκτή (microcontroller).

Η τρίτη κατηγορία αισθητήρων είναι το γυροσκόπιο. Σε όλες τις κινήσεις του νοηματιστή η κατεύθυνση και η κλίση του χεριού είναι απαραίτητες για να αποδοθεί το σωστό νόημα, καθώς ένα γράμμα μπορεί να έχει την ίδια μορφή (κλίση δαχτύλων) αλλά διαφορετική κλίση και κατεύθυνση χεριού. Για παράδειγμα, τα γράμματα «Z» και «Η» έχουν ίδια μορφή αλλά διαφορετική κλίση και κατεύθυνση. Οπότε τοποθετήσαμε το γυροσκόπιο στην πλακέτα (protoboard) που είναι τοποθετημένη στη ραχιαία επιφάνεια της παλάμης.

[4] Η εικόνα προέρχεται από Καραπάντζος, 2015

2.3.2 Arduino Uno Επεξεργασία σημάτων

Όλα τα αναλογικά σήματα που προέρχονται από τους αισθητήρες της κλίσης των δαχτύλων (flex sensors), της επαφής των δαχτύλων (contact sensors) και του γυροσκοπίου (gyroscope) πρέπει να μετατραπούν σε ψηφιακά, έτσι ώστε να επεξεργαστούν από τον μικροελεγκτή (microcontroller). Όλοι οι μικροελεγκτές (microcontrollers) έχουν αναλογικές εισόδους που μετατρέπουν το σήμα σε ψηφιακό. Στη δική μας περίπτωση το Arduino Uno διαθέτει έξι τέτοιες εισόδους (Arduino, 2017). Όμως εμείς στο σύνολο διαθέτουμε δεκατέσσερα αναλογικά σήματα (οκτώ από τους αισθητήρες κλίσης, τέσσερα από τους αισθητήρες επαφής και δύο από το γυροσκόπιο), με αποτέλεσμα να χρειαζόμαστε έναν μετατροπέα αναλογικού σήματος σε ψηφιακό. Αποφασίσαμε να χρησιμοποιήσουμε τον MCP3008 για το χαμηλό κόστος του αλλά και επειδή παρέχει 10 bits ανάλυση σε ταχύτητες 200kbps (Microchip, 2008). Δηλαδή παίρνει δύο μετρήσεις κάθε μιλισεκόντ, επομένως, εάν έχουμε είσοδο 5V θα έχουμε ευαισθησία 4mv για κάθε bit $\left(\frac{5V}{2^{10}}\right)$ (Microchip, 2008).

Όταν ο μικροελεγκτής (microcontroller) λάβει όλα τα σήματα από τους αισθητήρες, ο αλγόριθμος μπορεί να τα επεξεργαστεί για να αναγνωρίσει το γράμμα που έχει νοηματίσει ο νοηματιστής. Αρχικά κάθε γράμμα του αλφάβητου νοηματίζεται και σημειώνονται οι τιμές που επιστρέφουν οι αισθητήρες ενώ στη συνέχεια αυτές τοποθετούνται σε έναν πίνακα για να μπορέσει να τις χρησιμοποιήσει ο αλγόριθμος.

2.3.3 Σύστημα εξόδου

Ο μικροελεγκτής (microcontroller) τροφοδοτείται μέσω του καλωδίου usb που είναι συνδεδεμένο με τον ηλεκτρονικό υπολογιστή. Επίσης, μέσω του καλωδίου usb, επικοινωνεί ο μικροελεγκτής (microcontroller) με τον υπολογιστή. Μόλις αναγνωριστεί ένα γράμμα του αλφάβητου η πληροφορία αυτή μεταφέρεται μέσω του καλωδίου usb (Serial Communication) και εμφανίζεται στην οθόνη του υπολογιστή.

3. Μεθοδολογία

Για να δημιουργηθεί μια συσκευή για τη μετάφραση γραμμάτων του νοηματικού αλφάβητου σε μορφή που θα είναι κατανοητή από ανθρώπους που δεν μιλούν τη νοηματική γλώσσα, έπρεπε να μεσολαβήσουν τα εξής στάδια:

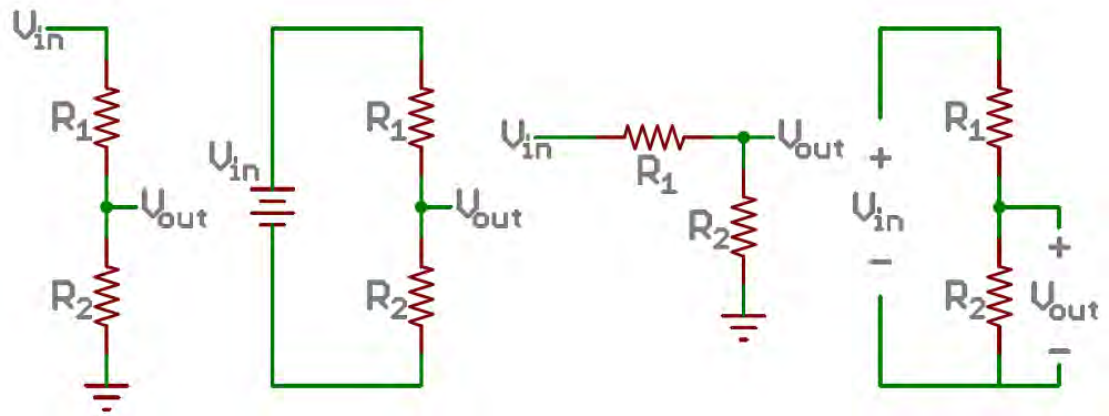
1. Επιλογή των κατάλληλων αισθητήρων καθώς και απόφαση σχετικά με το πώς και το πού θα τοποθετηθεί ο καθένας από αυτούς πάνω στο γάντι, ώστε να έχουμε όσο το δυνατόν καλύτερα αποτελέσματα.
2. Προετοιμασία των αισθητήρων (soldering) για να τοποθετηθούν (ράψιμο) στο γάντι.
3. Σχεδιασμός της πλακέτας (protoboard), που χρησιμεύει ως βάση για τον μικροελεγκτή (Arduino Uno), καθώς και τοποθέτηση των εξαρτημάτων (ADC, gyro, resistors και καλωδίων). Έτσι, θα ελαχιστοποιηθεί ο χώρος που χρειάζεται και θα γίνει συγκόλληση όλων των απαραίτητων εξαρτημάτων στη σωστή θέση για να μπορούν να μεταδίδονται τα σήματα από τους αισθητήρες στον μικροελεγκτή (microcontroller) και να επεξεργαστούν από αυτόν και από το software.
4. Ανάπτυξη του αναγκαίου κώδικα για την αναγνώριση των σημάτων που προέρχονται από τους αισθητήρες και την αναγνώριση κάποιων γραμμάτων.
5. Τελειοποίηση του κώδικα και προσπάθεια για να υπάρχουν όσο το δυνατόν καλύτερες προβλέψεις στην αναγνώριση ολόκληρου του αλφάβητου.

3.1 Ανάπτυξη υλικού

Το υλικό μας αποτελείται από τις τρεις κατηγορίες αισθητήρων (κλίσης, επαφής, γυροσκόπιο), τον μετατροπέα αναλογικού σε ψηφιακό σήμα, καθώς και τον μικροελεγκτή (Arduino UNO).

3.1.1 Διαχωριστής Τάσης (Voltage Divider)

Ο διαχωριστής τάσης είναι ένα κύκλωμα, το οποίο έχοντας είσοδο μια υψηλή τάση την μετατρέπει σε μια μικρότερη τάση (Wikipedia, 2017). Για να δημιουργήσουμε ένα τέτοιο κύκλωμα (Εικόνα 5) χρειαζόμαστε δύο αντιστάτες σε σειρά (R1 & R2) και μία είσοδο τάσης (Vin). Ως έξοδο (Vout) παίρνουμε μια μικρότερη τάση που εξαρτάται από την είσοδο και τους δύο αντιστάτες μας .



Εικόνα 5. Παραδείγματα διαχωριστών τάσεων

Η εξίσωση του διαχωριστή τάσης προϋποθέτει να είναι γνωστές τρεις τιμές:

1. Η είσοδος της τάσης
2. Ο πρώτος αντιστάτης
3. Ο δεύτερος αντιστάτης

Έχοντας αυτές τις τιμές μπορούμε να υπολογίσουμε την έξοδο της τάσης με την παρακάτω εξίσωση (Sparkfun, 2017):

$$Vout = Vin * \frac{R2}{R1 + R2}$$

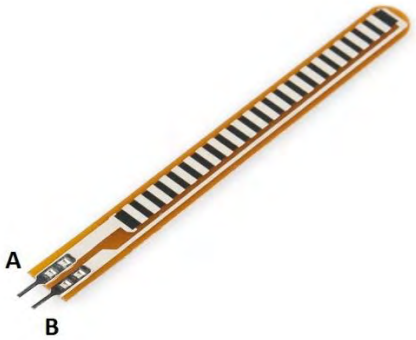
Εξίσωση 1 Υπολογισμός τάσης εξόδου

Πολλοί αισθητήρες, όπως και ο flex στη δική μας περίπτωση, για να μπορέσουν να ενσωματωθούν με μικροελεγκτές (microcontrollers), είναι αναγκαίο να έχουν έναν διαχωριστή τάσης.

3.1.2 Flex sensors

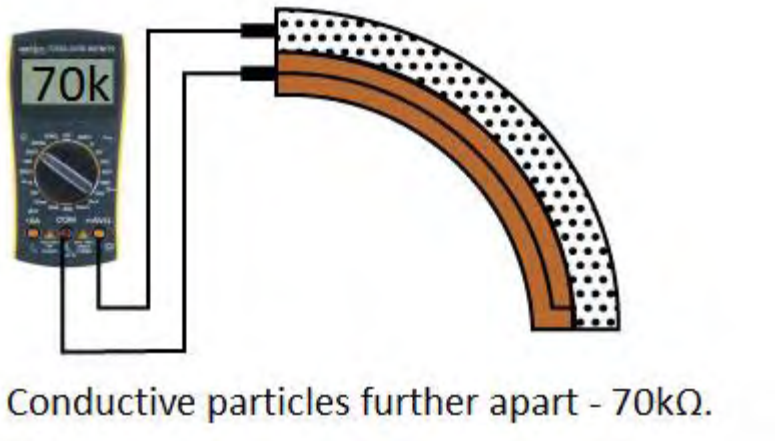
Μετά από έρευνα και αναζήτηση παρόμοιων συσκευών αποφασίστηκε ότι ο καλύτερος τρόπος για να μετρηθεί η κλίση των δαχτύλων είναι με τους Flex Sensors. Αυτού του είδος οι αισθητήρες έχουν δύο μεγέθη: 2.2 ίντσες (5.588 εκ.) και 4.5 ίντσες (11.43 εκ.).

Αυτοί οι αισθητήρες, αν τοποθετηθούν οριζόντια με 0^o κλίση, θα μοιάζουν με έναν αντιστάτη 30 kΩ. Όσο αυξάνουμε την κλίση (0^o - $+90^o$) και ο αισθητήρας λυγίζει, η αντίσταση μεταξύ των δύο ακροδεκτών του αισθητήρα (Εικόνα 6) αυξάνεται και μπορεί να φτάσει ως 70 kΩ στις 90^o μοίρες (SpectraSymbol, 2017).



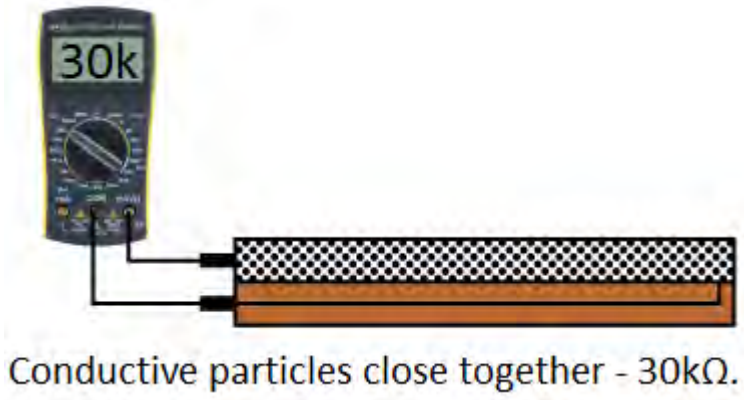
Εικόνα 6. Αισθητήρας κλίσης (flex sensor)

Ο αισθητήρας από τη μία πλευρά είναι καλυμμένος με πολυμερές μελάνι (polymer ink) το οποίο έχει μέσα του σωματίδια που είναι καλοί αγωγοί του ηλεκτρισμού. Όταν ο αισθητήρας έχει 0^o κλίση, τα σωματίδια δίνουν στο μελάνι αντίσταση της τάξης των 30kΩ. Όμως, όταν αυξάνεται η κλίση του αισθητήρα, τα σωματίδια απομακρύνονται μεταξύ τους με αποτέλεσμα να αυξάνεται η αντίσταση (50kΩ - 70kΩ στις 90^o, όπως φαίνεται στην Εικόνα 7) (Sparkfun, 2017).



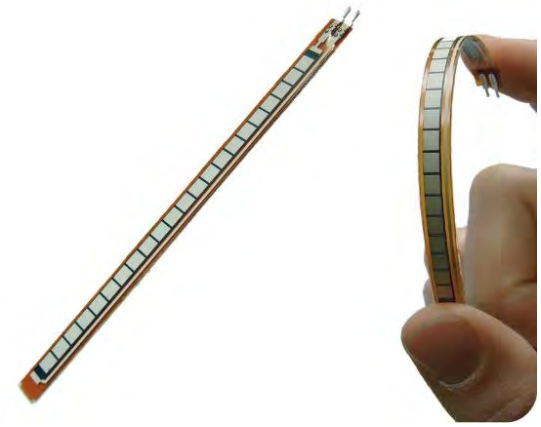
Εικόνα 7. Αισθητήρας κλίσης 90^{o}

Όταν ο αισθητήρας ξαναγυρίζει στις 0^{o} κλίση, η αντίσταση επιστρέφει πάλι στα 30kΩ (Εικόνα 8).



Εικόνα 8. Αισθητήρας κλίσης 0^{o}

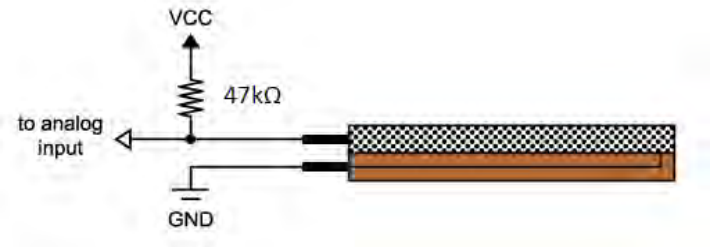
Επίσης, ο αισθητήρας είναι σχεδιασμένος να λυγίζει μόνο προς μία κατεύθυνση (Εικόνα 9). Λυγίζοντας τον αισθητήρα προς την αντίθετη κατεύθυνση δεν θα μας δώσει σωστά δεδομένα, καθώς υπάρχει περίπτωση να υποστεί βλάβη. Ακόμα, πρέπει να δοθεί προσοχή ώστε να μην λυγίζει ο αισθητήρας μέχρι τη βάση, διότι υπάρχει περίπτωση να σπάσει (Sparkfun, 2017).



Εικόνα 9. Ευλυγισία αισθητήρα

Συνδυάζοντας, λοιπόν, τον αισθητήρα με έναν αντιστάτη σταθερού μεγέθους, με σκοπό να δημιουργήσουμε έναν διαχωριστή τάσης (voltage divider), δημιουργείται μια τάση που μπορεί να αναγνωριστεί από έναν μικροελεγκτή (microcontroller), όπως το Arduino, και να μετατραπεί το σήμα από αναλογικό σε ψηφιακό, έτσι ώστε να μπορέσει να χρησιμοποιηθεί από το πρόγραμμα στη συνέχεια.

Ο πιο εύκολος τρόπος να ενσωματώσουμε τον αισθητήρα στο κύκλωμα είναι με τον διαχωριστή τάσης. Το κύκλωμα (Εικόνα 10) χρειάζεται έναν αντιστάτη με αντίσταση μεταξύ 10kΩ και 100kΩ. Συνήθως χρησιμοποιείται μια τιμή αντίστασης μεταξύ της μικρότερης και της μεγαλύτερης τιμής που μπορεί να πάρει ο αισθητήρας μας.



Εικόνα 10. Παράδειγμα συνδεσμολογίας αισθητήρα κλίσης

Κλίση αισθητήρα	**R_2 Αισθητήρας**	**R_1 Σταθερός**	$\frac{R2}{R1+R2}$	**V_{out}**

0^o	30kΩ	47kΩ	0.389	1.94
90^o	70kΩ	47kΩ	0.875	2.99

Πίνακας 1. Υπολογισμός Vout ενός αισθητήρα κλίσης με Vin 5V

3.1.3 Αισθητήρας Πίεσης (Force sensitive sensor)

Οι αισθητήρες fsr402, που χρησιμοποιήσαμε, είναι σχεδιασμένοι για να αναγνωρίζουν την πίεση κυρίως μεταξύ των δαχτύλων του ανθρώπου σε τομείς όπως αυτός της αυτοκινητοβιομηχανίας, της ιατρικής, της βιομηχανίας, της ρομποτικής κ.ά. (Interlink Electronics, 2017).

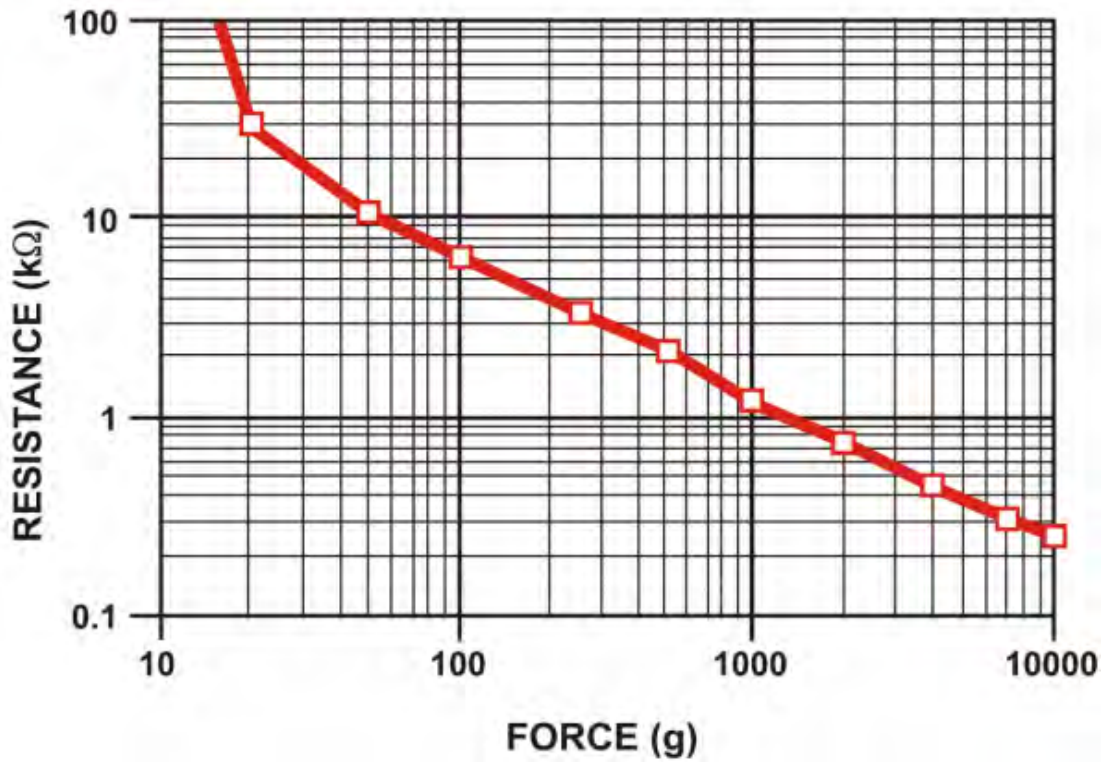
Οι αισθητήρες πίεσης είναι σχεδιασμένοι να μετρούν την ύπαρξη και το σχετικό μέγεθος της πίεσης σε μια περιοχή. Η αντίσταση του αισθητήρα αυξάνεται και μειώνεται ενώ εξαρτάται από τη δύναμη που του ασκείται. Όταν δεν ασκείται καθόλου πίεση στον αισθητήρα, η αντίσταση του είναι μεγαλύτερη από 1ΜΩ. Όσο αυξάνουμε την πίεση που ασκείται στον αισθητήρα, τόσο η αντίσταση μειώνεται μεταξύ των δύο ακροδεκτών του αισθητήρα (Sparkfun, 2017). Με τη χρήση ενός δεύτερου στατικού αντιστάτη μπορούμε να δημιουργήσουμε έναν διαχωριστή τάσης για να μπορέσει ο μικροελεγκτής (microcontroller) να διαβάσει τις τιμές του αισθητήρα.

Οι αισθητήρες αποτελούνται από πολυμερές μελάνι που είναι καλός αγωγός του ηλεκτρισμού. Η περιοχή που αναγνωρίζει την πίεση που της ασκείται αποτελείται από δύο ειδών σωματίδια: τα σωματίδια που είναι καλοί αγωγοί του ηλεκτρισμού και τα σωματίδια που είναι κακοί αγωγοί του ηλεκτρισμού. Όταν ασκείται πίεση, τα σωματίδια έρχονται πιο κοντά μεταξύ τους, με αποτέλεσμα να αλλάζει η αντίσταση μεταξύ των ακροδεκτών του αισθητήρα (Wikipedia, 2017). Το μειονέκτημα του αισθητήρα είναι ότι μπορεί η διαφορά μεταξύ των μετρήσεων να είναι μεγαλύτερη του 10%.

Εικόνα 11. Αισθητήρας επαφής (contact sensor)

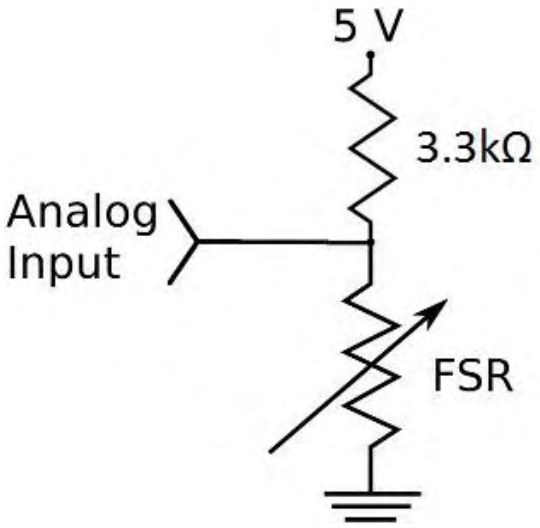
Επίσης, ένα χαρακτηριστικό που έχουν αυτοί οι αισθητήρες είναι ότι μπορεί να ρυθμιστεί το διάστημα που ο αισθητήρας αντιλαμβάνεται την πίεση. Δηλαδή, το διάστημα είναι η μικρότερη και η μέγιστη τιμή πίεσης που ο αισθητήρας μπορεί να αντιληφθεί. Όσο μικρότερο είναι το διάστημα, τόσο πιο ευαίσθητος και ακριβής θα είναι στις μετρήσεις. Οποιαδήποτε πίεση εκτός του διαστήματος δεν θα μπορεί να μετρηθεί και υπάρχει η πιθανότητα να καταστραφεί ο αισθητήρας. Ο αισθητήρας που χρησιμοποιούμε εμείς μπορεί να αντιληφθεί από 100 γραμμάρια έως 10 κιλά, αλλά δεν μπορεί να καταλάβει τη διαφορά μεταξύ των 11 κιλών και των 20 κιλών.

Στο παρακάτω διάγραμμα παρατηρούμε τη σχέση μεταξύ της πίεσης και της αντίστασης. Η σχέση μεταξύ των δύο είναι κυρίως γραμμική από τα 50 γραμμάρια και πάνω. Όμως, για να ενεργοποιηθεί ο αισθητήρας απαιτείται μια ελάχιστη τιμή δύναμης, περίπου 100 γραμμάρια, ώστε να γίνει γραμμική η αντίσταση και να πέσει κάτω από τα 10kΩ.



Εικόνα 12. Διάγραμμα σχέσης πίεσης-δύναμης

Ο πιο εύκολος τρόπος να ενσωματώσουμε τον αισθητήρα στο κύκλωμα είναι με τον διαχωριστή τάσης. Το κύκλωμα (Εικόνα 13) χρειάζεται έναν αντιστάτη με αντίσταση 3.3 kΩ.



Εικόνα 13. Παράδειγμα συνδεσμολογίας αισθητήρα επαφής

3.1.4 SPI

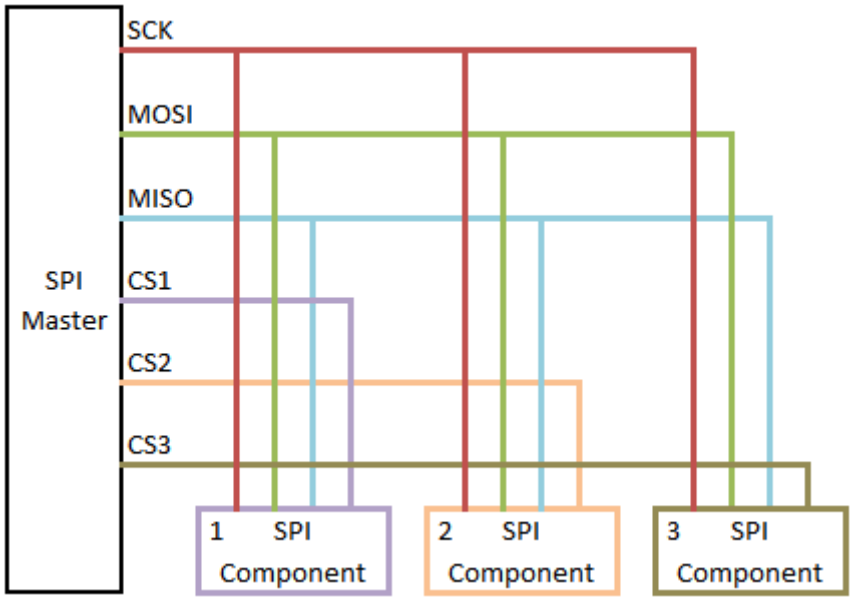
Το SPI, ή αλλιώς στα ελληνικά περιφερειακή σειριακή διεπαφή, είναι ένα πρωτόκολλο σύγχρονης σειριακής επικοινωνίας μεταξύ ενσωματωμένων συστημάτων σε κοντινές αποστάσεις. Η διεπαφή αναπτύχθηκε από τη Motorola στα τέλη της δεκαετίας του ΄80 (Wikipedia, 2017).

Το SPI πρωτόκολλο είναι ικανό να τρέξει σε εφτά διαφορετικές ταχύτητες από τα 62.5kHz μέχρι τα 8MHz. Όλα τα μοντέλα SPI αποτελούνται από έναν μόνο master (αρχηγό) και έναν ή περισσότερους slaves (σκλάβους). Υπάρχει μόνο ένας master διότι πάνω στο SPI bus μόνο αυτός έχει τη δυνατότητα να ελέγχει το σήμα του ρολογιού. Για να ξεκινήσει η επικοινωνία μεταξύ τους αρκεί μόνο μία από τις γραμμές (pin) να αλλαχθεί σε logic low state (χαμηλή λογική κατάσταση) (Hienzsch, Rheingold Heavy, 2015). Το πλεονέκτημα του SPI είναι ότι έχει διαφορετικές γραμμές επικοινωνίας για την αποστολή και την παραλαβή δεδομένων, με αποτέλεσμα να μπορεί να υποστηρίξει την ταυτόχρονη αμφίδρομη μεταφορά δεδομένων μεταξύ του master και του slave.

Η βασική αρχιτεκτονική SPI περιλαμβάνει το ελάχιστο τρεις γραμμές μεταξύ του master και των slaves:

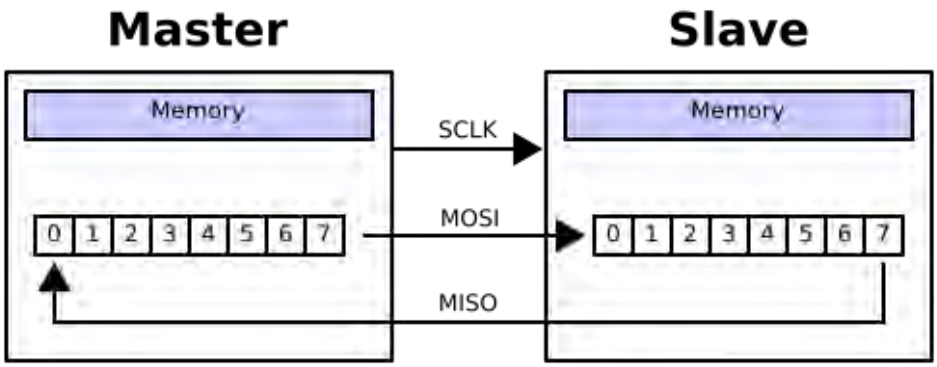
1. Το ρολόι που ονομάζεται SCK
2. Τη γραμμή Master IN/Slave Out, που ονομάζεται MISO (Me-So)
3. Τη γραμμή Master Out/Slave, που ονομάζεται MOSI (Moh-See)

Συνήθως υπάρχει και η γραμμή Slave Select (SS) η οποία ελέγχει με ποιον slave γίνεται η επικοινωνία εκείνη τη στιγμή.



Εικόνα 14. Παράδειγμα επικοινωνίας συσκευών SPI

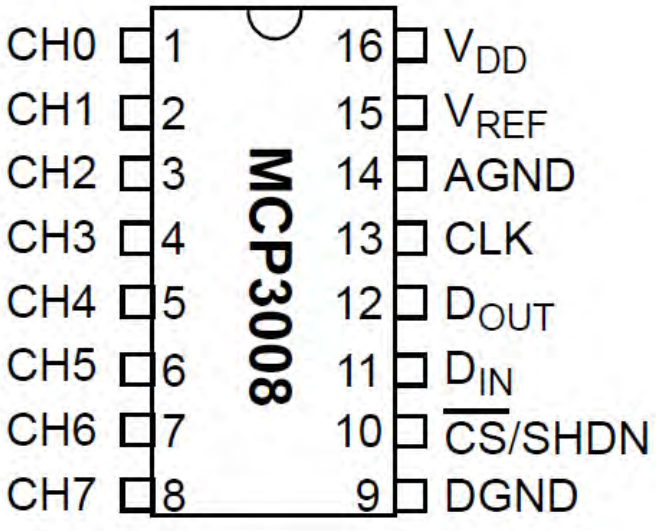
Για να ξεκινήσει η επικοινωνία μεταξύ τους, ο master ρυθμίζει το ρολόι χρησιμοποιώντας μια συχνότητα που υποστηρίζεται και από τον master και από τον slave. Μετά ο master επιλέγει τη συσκευή slave αλλάζοντας την τιμή της γραμμή SS (slave select line) σε logic low level state. Κατά τη διάρκεια ενός κύκλου ρολογιού, ο master στέλνει ένα bit στη γραμμή Mosi και ο slave το διαβάζει, ενώ ταυτόχρονα ο slave στέλνει ένα bit στη γραμμή Miso και ο master το διαβάζει. Συνήθως η επικοινωνία περιλαμβάνει 8 bits λέξεις, μία λέξη στον master και μία λέξη στον slave. Υπάρχουν δύο καταχωρητές, ένας στον master και ένας στον slave. Συνήθως τα δεδομένα που στέλνονται από τον master ολισθαίνουν από το πιο σημαντικό bit πρώτα, καθώς ταυτόχρονα λαμβάνεται το λιγότερο σημαντικό bit στον ίδιο καταχωρητή. Στον άλλο καταχωρητή γίνεται το αντίθετο: ολισθαίνει το λιγότερο σημαντικό bit και ταυτόχρονα λαμβάνεται και ολισθαίνει το πιο σημαντικό bit από τον άλλο καταχωρητή.



Εικόνα 15. Παράδειγμα μεταφοράς δεδομένων μεταξύ συσκευών SPI

3.1.5 MCP3008 Analog-to-digital converter

Το MCP3008 έχει οχτώ κανάλια με 10 bits ανάλυση και επικοινωνεί μέσω του SPI. Αφού έχει οχτώ κανάλια είναι ικανό να λαμβάνει και να επεξεργάζεται οχτώ διαφορετικές τάσεις (voltages) (Microchip, 2008). Αυτές οι τιμές μπορούν να διαβαστούν με δύο τρόπους: ο πιο τυπικός τρόπος ονομάζεται “single ended input”, κατά τον οποίο το επίπεδο της τάσης μετριέται με τη βοήθεια της τάσης αναφοράς (V_{REF}). Ο άλλος τρόπος είναι να μετρηθεί η διαφορά μεταξύ των IN+ και IN-.



Εικόνα 16. Σχεδιάγραμμα MCP3008

Τα 10 bits ανάλυσης αναφέρονται στο διάστημα ανάλυσης του δείγματος που μπορεί να πάρει το ADC (analog-digital-converter), δηλαδή, αν B0000000000 = 0 και B1111111111 = 1023, τότε μπορούμε να μετρήσουμε 1024 διαφορετικά επίπεδα τάσης μεταξύ της γείωσης (Ground) και της τάσης αναφοράς (V_{REF}) (Dicola, 2016). Οπότε, αν έχουμε τάση αναφοράς 5V και μπορούμε αυτή την τάση να την χωρίσουμε 1024 φορές, κάθε βήμα θα είναι 0.0049V. Εάν έχουμε 3.3V, τότε κάθε βήμα θα είναι 0.0032V. Μπορούμε να έχουμε και μικρότερα βήματα, όμως από ένα σημείο και μετά υπάρχει κίνδυνος να προκύψουν λάθη στις μετρήσεις μας.

Την τάση αναφοράς (V_{REF}) τη χρησιμοποιούμε για να συγκρίνουμε την τάση που ελέγχουμε. Για παράδειγμα, είναι σαν να κάνουμε την ερώτηση «Πόσες φορές μικρότερη ή περισσότερη είναι η τάση που ελέγχουμε σε σχέση με την τάση αναφοράς;».

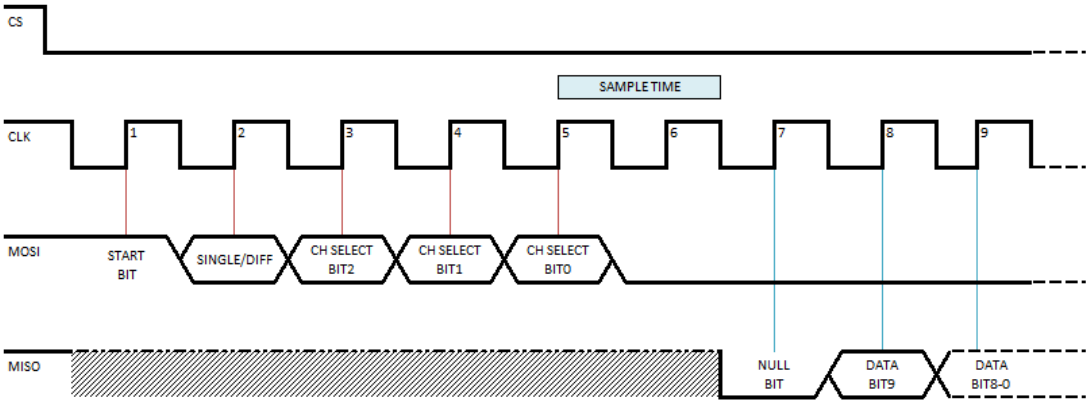
Ο συγχρονισμός της μετάδοσης των δεδομένων είναι σημαντικός, διότι χρειάζεται χρόνος για να γίνει η μέτρηση πριν μπορέσουμε να τη διαβάσουμε. Η διαδικασία που εκτελείται είναι η εξής:

1. Λέμε στο MCP3008 να ετοιμαστεί για μέτρηση.
2. Λέμε σε ποιο κανάλι θέλουμε να κάνουμε τη μέτρηση.
3. Περιμένουμε τον ελάχιστο χρόνο που χρειάζεται για να γίνει η μέτρηση.
4. Ξεκινάμε να διαβάζουμε τα δεδομένα της μέτρησης.

Για να γίνει η επικοινωνία μεταξύ του MCP3008 και του Arduino στέλνονται σε κάθε χτύπο ρολογιού 8 bits δεδομένων. Για να επιτευχθεί ανάλυση 10 bits χρειαζόμαστε:

1. Ένα bit εκκίνησης
2. Ένα bit single/diff bit
3. Τρία bits επιλογής καναλιού
4. Αναμονή ενός κύκλου ρολογιού
5. Ένα null bit
6. Δέκα bits δεδομένων

Οπότε συνολικά χρειαζόμαστε δεκαεφτά κύκλους ρολογιού.



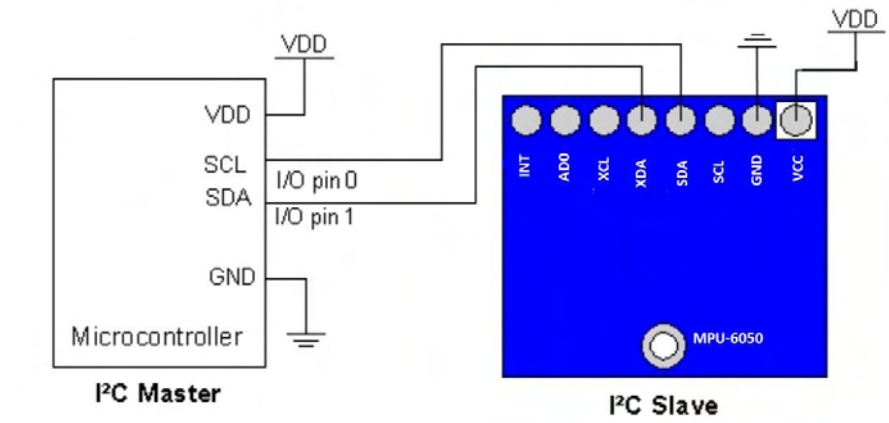
Εικόνα 17. Παράδειγμα επικοινωνίας

3.1.6 MPU-6050 Γυροσκόπιο/Επιταχυνσιόμετρο

Το MPU-6050 περιλαμβάνει γυροσκόπιο και επιταχυνσιόμετρο. Είναι αρκετά ακριβές καθώς διαθέτει 16 bits μετατροπέα από αναλογικό σε ψηφιακό σήμα για το κάθε κανάλι (Sparkfun, 2017). Για αυτό το λόγο μπορεί να μετρήσει τις συντεταγμένες x, y και z ταυτόχρονα. Η επικοινωνία με το Arduino επιτυγχάνεται μέσω του πρωτόκολλου I^2C. Το MPU-6050 διαθέτει οχτώ γραμμές (pins):

1. Vcc: είσοδος τάσης για να λειτουργήσει.
2. GND: γείωση
3. SCL: γραμμή του ρολογιού. Χρησιμοποιείται για τον συγχρονισμό των μεταφορών δεδομένων μέσω του I2C.
4. SDA: γραμμή μεταφοράς δεδομένων
5. XDA: επιπρόσθετη γραμμή για την προσθήκη μαγνητόμετρου
6. XCL: επιπρόσθετη γραμμή για την προσθήκη μαγνητόμετρου
7. AD0: επιπρόσθετη γραμμή για την προσθήκη μαγνητόμετρου
8. INT: γραμμή για την αποστολή σήματος διακοπής

Για τη σύνδεσή του με το Arduino χρειαζόμαστε δύο αναλογικές εισόδους, ρεύμα, γείωση και μία ψηφιακή είσοδο/έξοδο για την αποστολή του σήματος διακοπής, αν θελήσουμε (Sanjeev, 2016).



Εικόνα 18. Παράδειγμα συνδεσμολογίας του MPU-6050

Αφού το γυροσκόπιο χρησιμοποιεί I2C για την επικοινωνία, χρησιμοποιήσαμε τις αναλογικές εισόδους A4 και A5 του Arduino που είναι σχεδιασμένες να λειτουργούν έτσι χρησιμοποιώντας την Wire βιβλιοθήκη.

3.1.7 Arduino

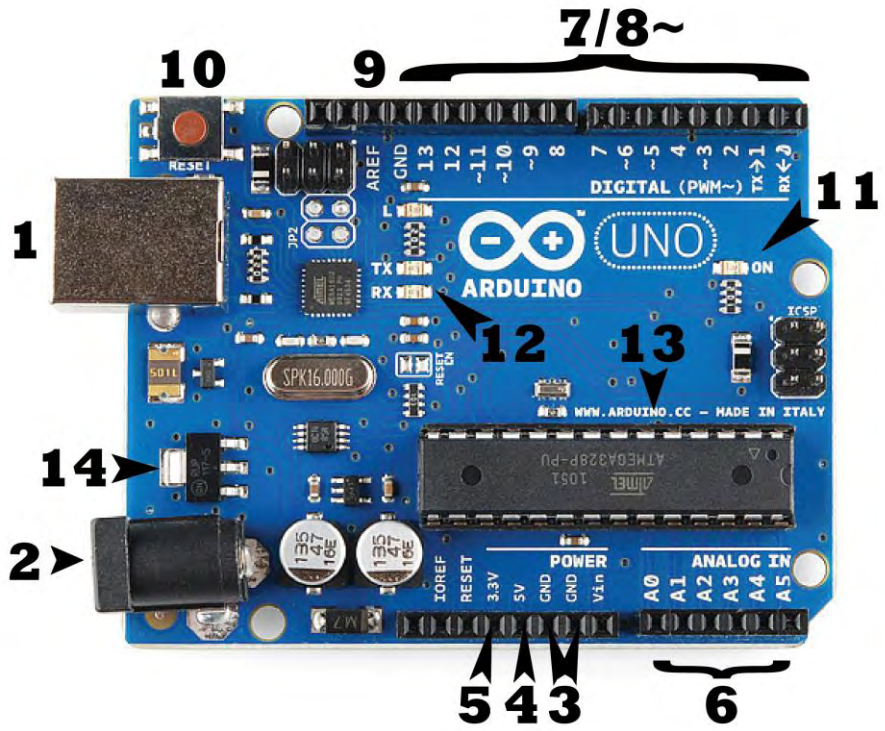
Το Arduino είναι μια ανοιχτή πλατφόρμα που χρησιμοποιείται για τη δημιουργία ηλεκτρονικών πρότζεκτς. Το Arduino χωρίζεται σε δύο κατηγορίες, η μία κατηγορία είναι το φυσικό κομμάτι (hardware), που είναι ο μικροελεγκτής (microcontroller), και η δεύτερη κατηγορία είναι το πρόγραμμα (IDE) που «τρέχει» στον υπολογιστή για να γραφτεί ο αλγόριθμος/πρόγραμμα που θα ανέβει στο Arduino για να εκτελέσει συγκεκριμένες λειτουργίες. Το Arduino μπορεί να συνδεθεί με κουμπιά, LEDs, ηχεία, GPS, internet, ρελέ, αισθητήρες, κ.ά. (Arduino, 2017).

3.1.8 Arduino Uno

Στο γάντι χρησιμοποιήσαμε το Arduino Uno. Αυτό αποτελείται από (Arduino, 2017):

1. Θύρα Usb, η οποία έχει διπλή χρησιμότητα, είτε ως πηγή ρεύματος είτε για να ανεβάσει ο χρήστης τον κώδικα που δημιούργησε στο Arduino.

2. Θύρα ρεύματος, το Arduino έχει από τις πιο συνηθισμένες θύρες τροφοδοσίας ρεύματος που χρειάζεται τροφοδοτικό 2.1mm μεταξύ των 6 και 12 V DC.
3. GND, είναι η γείωση που είναι απαραίτητη σε όλα τα κυκλώματα.
4. 5V, δίνει 5V τάση για να τροφοδοτήσει ο χρήστης το κύκλωμά του.
5. 3.3V, είναι το ίδιο με τα 5V απλά σε μικρότερη τάση.
6. Έξι αναλογικές εισόδους (A0-A5) οι οποίες μπορούν, για παράδειγμα, να διαβάσουν μια τιμή από έναν αισθητήρα και να τη μετατρέψουν σε ψηφιακή.
7. Δεκατέσσερις ψηφιακές εισόδους/εξόδους (0-13) που χρησιμεύουν, για παράδειγμα, είτε για να διαβάσουν μια τιμή (εάν πατήθηκε ένα κουμπί) είτε ως έξοδος σήματος (να ανάψουν ένα LED).
8. Κάποιες από τις δεκατέσσερις ψηφιακές εισόδους/εξόδους έχουν ένα «~» (οι 3, 5, 6, 9, 10 και 11). Αυτές είναι τύπου PWM (Pulse With Modification) και μπορούν είτε να χρησιμεύσουν για τη διαχείριση ενός RGB LED (μπορεί να πάρει όλους τους συνδυασμούς χρωμάτων που υπάρχουν στο RGB) ή να ελέγξουν μία κατεύθυνση σε ένα servo motor.
9. Μερικές φορές χρειάζεται μια τάση αναφοράς που δρα ως το άνω όριο για τις αναλογικές εισόδους.
10. Reset, χρησιμεύει όταν θέλουμε να κάνουμε επανεκκίνηση στο Arduino και να ξεκινήσει η λειτουργία του από την αρχή. Λειτουργεί συνδέοντας τη γείωση (GND) με το reset του Arduino.
11. Ένα LED που ανάβει μόνο όταν το Arduino Uno είναι συνδεδεμένο σε κάποια πηγή ρεύματος είτε μέσω του usb είτε της τροφοδοσίας.
12. Δύο LEDs, TX και RX. Όταν ανάβουν μας ενημερώνουν ότι το Arduino είτε αποστέλλει είτε λαμβάνει δεδομένα σειριακά.
13. Ενσωματωμένο κύκλωμα, στην περίπτωση μας είναι το AtMega328.
14. Σταθεροποιητής τάσης, είναι υπεύθυνος να σταθεροποιεί την τάση και να τροφοδοτεί το Arduino.



Εικόνα 19. Arduino Uno

3.1.9 Μνήμη Arduino

Το Arduino Uno έχει τρία είδη μνήμης (Arduino, 2017):

1. Flash memory, είναι η μνήμη στην οποία αποθηκεύεται όλο το πρόγραμμα (κώδικας).
2. SRAM (Static Random Access Memory), είναι η μνήμη με την οποία το πρόγραμμα δημιουργεί και επεξεργάζεται μεταβλητές, όταν «τρέχει» ο κώδικας.
3. EEPROM (Electrically Erasable Programmable Read-Only Memory), είναι η μνήμη που μπορεί να χρησιμοποιήσει ο προγραμματιστής για να αποθηκεύσει δεδομένα που δεν αλλάζουν συχνά.

Η πρώτη και η τρίτη μνήμη δεν χάνονται, δηλαδή, ακόμα και χωρίς ρεύμα, τα δεδομένα τους θα διατηρηθούν, ενώ στη δεύτερη μνήμη τα δεδομένα χάνονται κάθε φορά που διακόπτεται το ρεύμα στην πλακέτα.

Στο Arduino Uno το μέγεθος της Flash Memory είναι 32k bytes όπου τα 5k bytes χρησιμοποιούνται από τον bootloader, η SRAM είναι 2k bytes και η EEPROM είναι 1k byte.

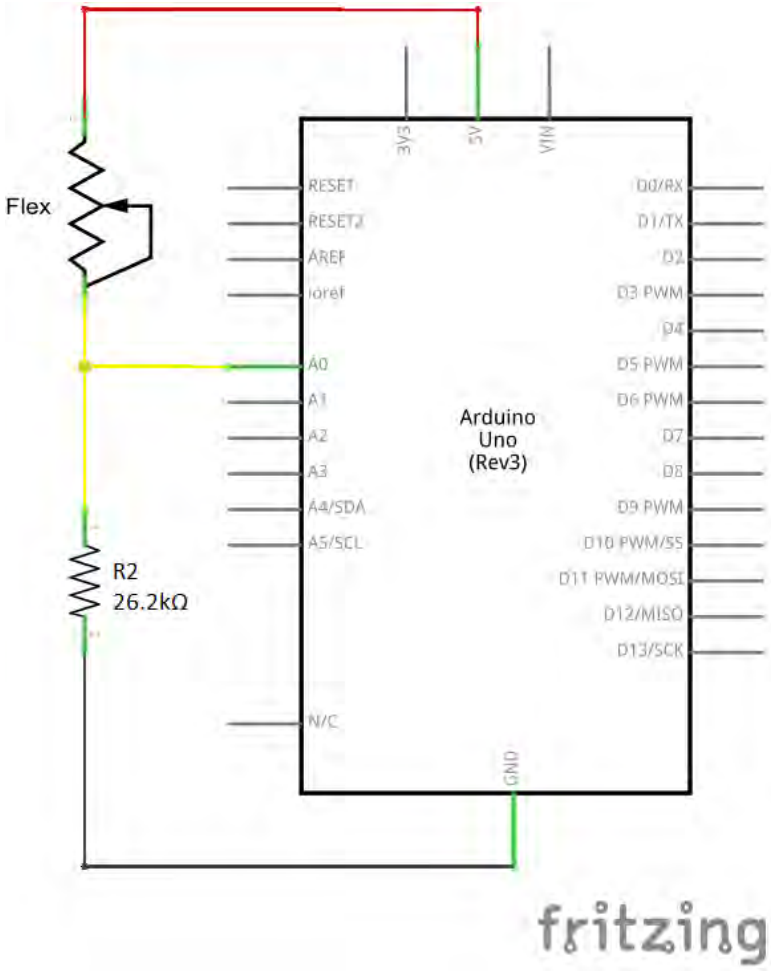
Το μέγεθος της SRAM είναι πολύ περιορισμένο στο Arduino Uno και, αν το πρόγραμμα την χρησιμοποιεί όλη, μπορεί να εμφανίσει προβλήματα δυσλειτουργίας, όπως να «τρέχει» περίεργα ο κώδικας ή να μην «τρέχει» καθόλου.

3.2 Κατασκευή της συσκευής

Η όλη διαδικασία της κατασκευής, της συναρμολόγησης της συσκευής και της επιλογής των διάφορων εξαρτημάτων, έγινε σε στάδια.

3.2.1 Δοκιμή αισθητήρων

Στο πρώτο στάδιο έπρεπε να ελεγχθεί η καλή λειτουργία όλων των αισθητήρων και η σωστή συνδεσμολογία. Οι πρώτες δοκιμές έγιναν με έναν αισθητήρα κλίσης και έναν αισθητήρα επαφής. Αρχικά, οι δύο αισθητήρες τοποθετήθηκαν σε ένα breadboard. Η συνδεσμολογία που ακολουθήσαμε (Εικόνα 20) για τον αισθητήρα κλίσης ήταν να συνδέσουμε τον έναν ακροδέκτη του αισθητήρα στα 5V του Arduino, τον άλλο ακροδέκτη στην αναλογική είσοδο του Arduino, και, με έναν αντιστάτη (47kΩ), τον δεύτερο ακροδέκτη στη γείωση (GND) του Arduino.



Εικόνα 20. Συνδεσμολογία αισθητήρα κλίσης

Αφού παρατηρήσαμε ότι χρησιμοποιούσαμε τη σωστή συνδεσμολογία και ότι τα αποτελέσματα που επέστρεφαν οι αισθητήρες ήταν λογικά, συνεχίσαμε να δοκιμάζουμε τους υπόλοιπους αισθητήρες για να σιγουρευτούμε ότι δεν υπάρχει κάποιος ελαττωματικός που θα μας επέστρεφε εσφαλμένα αποτελέσματα. Το μόνο που χρειαζόταν στον κώδικα του Arduino ήταν να διαβάσουμε την τιμή που μας επέστρεφε ο αισθητήρας. Για να μπορέσουμε να ελέγξουμε αν οι τιμές που λαμβάναμε ήταν λογικές, έπρεπε να υπολογίσουμε την αντίσταση που μας επέστρεφε ο αισθητήρας. Την αντίσταση μπορούμε να την υπολογίσουμε από τον τύπο:

$$Resistance = R2 * \frac{VCC}{\left(\left(\frac{flexAnalogInput * VCC}{1023.0}\right) - 1.0\right)}$$

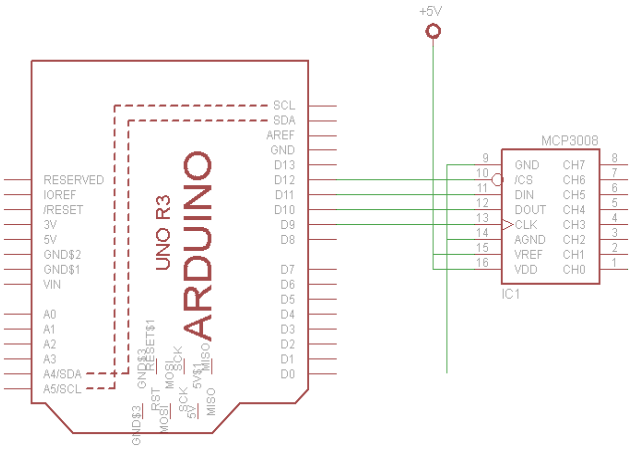
Εξίσωση 2. Υπολογισμός αντίστασης αισθητήρα κλίσης

Το R2 είναι ο αντιστάτης που χρησιμοποιήσαμε για να δημιουργήσουμε τον διαχωριστή τάσης, ο οποίος μετρήθηκε με πολύμετρο και βρέθηκε ότι η τιμή του ήταν 26.2kΩ, και το VCC είναι η τάση που παρέχεται από το Arduino, η οποία επίσης μετρήθηκε με πολύμετρο και ήταν 4.98V.

Στη συνέχεια δοκιμάσαμε τον μετατροπέα αναλογικού σήματος σε ψηφιακό. Αποφασίσαμε να συνδέσουμε δύο αισθητήρες κλίσης με τον ίδιο τρόπο, όπως είχαμε δοκιμάσει πριν, με τη διαφορά ότι, αντί να συνδέσουμε τον δεύτερο ακροδέκτη στις αναλογικές εισόδους του Arduino, τον συνδέσαμε σε δύο κανάλια από τον μετατροπέα μας. Η συνδεσμολογία που ακολουθήσαμε για τον μετατροπέα ήταν η εξής:

MCP3008	Arduino Uno
Pin 9: DGND	Ground
Pin 10: CS	D12
Pin 11: Din	D11
Pin 12: Dout	D10
Pin 13: CLK	D9
Pin 14: AGND	Ground
Pin 15: Vref	+5V
Pin 16: Vdd	+5V

Πίνακας 2. Συνδεσμολογία MCP3008



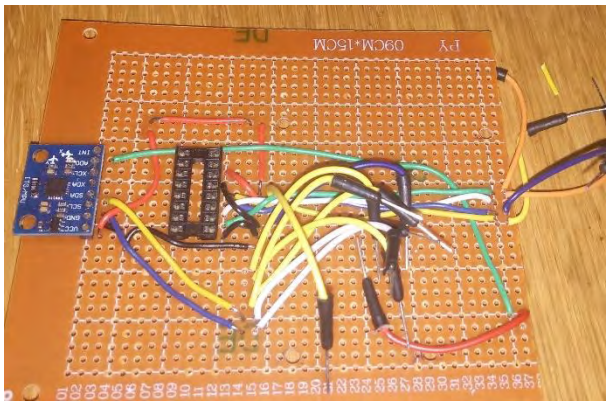
Εικόνα 21. Συνδεσμολογία γυροσκοπίου

Για να μπορέσουμε να πάρουμε τις τιμές από τους αισθητήρες, έπρεπε να εισαγάγουμε στον κώδικά μας τη βιβλιοθήκη MCP3008 καθώς και να δημιουργήσουμε ένα αντικείμενο από τη βιβλιοθήκη, βάζοντας ως παραμέτρους τις ψηφιακές εισόδους/εξόδους που χρησιμοποιήσαμε (D9 - D12).

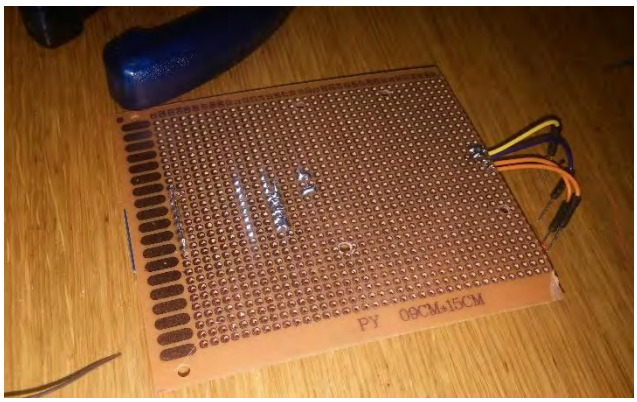
Τέλος, έπρεπε να δοκιμάσουμε το γυροσκόπιο. Η συνδεσμολογία που ακολουθήσαμε είναι η ίδια που αναφέρουμε στο Κεφάλαιο 3, όπου SDA και SCL είναι οι αναλογικές είσοδοι A4 και A5 αντίστοιχα. Επίσης, χρησιμοποιήσαμε τη βιβλιοθήκη MPU6050 και «τρέξαμε» ένα από τα έτοιμα παραδείγματα που μας παρείχε η βιβλιοθήκη.

3.2.2 Συναρμολόγηση της συσκευής

Στο δεύτερο στάδιο έπρεπε να κολλήσουμε προεκτάσεις καλωδίων σε όλους τους αισθητήρες κλίσης και επαφής. Αφού κολλήσαμε όλα τα απαραίτητα καλώδια, κόψαμε την πλακέτα (protoboard) σε μικρότερο μέγεθος για να είναι πιο βολική πάνω στο γάντι. Έπειτα αρχίσαμε να κολλάμε όλα τα απαραίτητα καλώδια, το γυροσκόπιο και τον μετατροπέα (ADC). Για να στερεώσουμε το Arduino στην

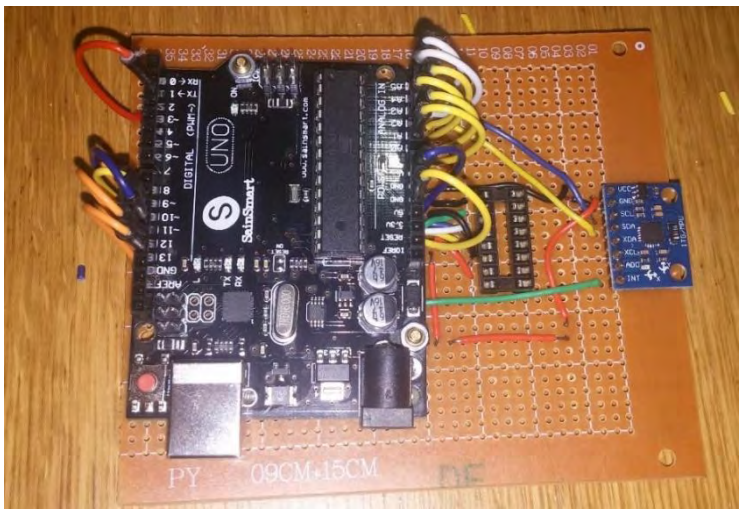


Εικόνα 22. Πρόοδος Κατασκευής #1



Εικόνα 23. Πρόοδος Κατασκευής #2

πλακέτα (protoboard) κάναμε τέσσερις τρύπες και το βιδώσαμε σε αυτό.

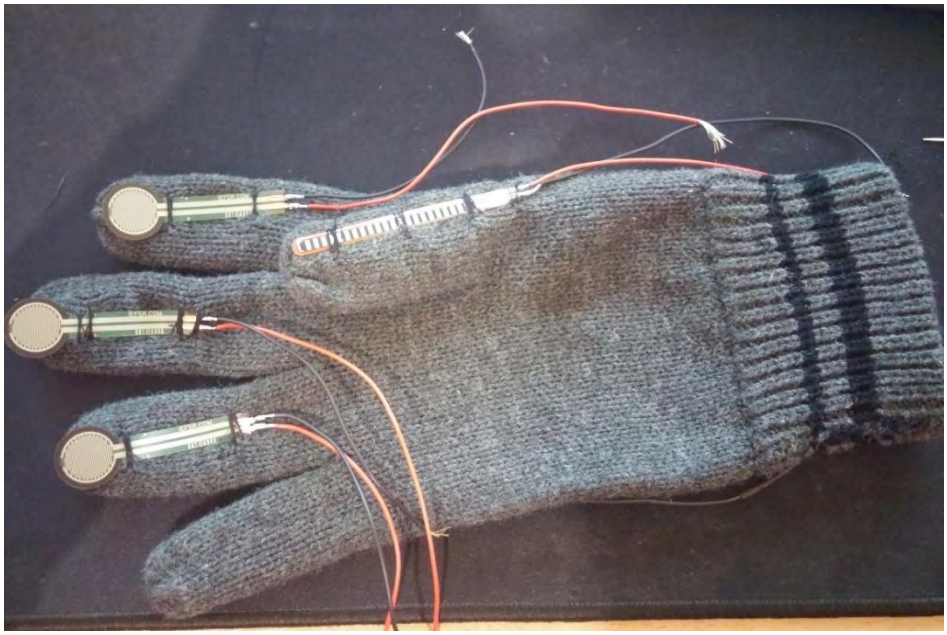


Εικόνα 24. Πρόοδος Κατασκευής #3

Αφού τοποθετήθηκαν και κολλήθηκαν όλα πάνω στην πλακέτα (protoboard), ξεκίνησε η τοποθέτηση και το ράψιμο των αισθητήρων στο γάντι.

3.2.3 Επιλογή γαντιού

Είχαμε δύο επιλογές όσον αφορά στην επιλογή του γαντιού. Η πρώτη επιλογή ήταν ένα υφασμάτινο γάντι πανομοιότυπο με αυτά που χρησιμοποιούνται στα χιόνια ενώ η δεύτερη επιλογή ήταν ένα πλαστικό γάντι που χρησιμοποιείται κυρίως σε δουλειές κουζίνας. Στο υφασμάτινο γάντι οι αισθητήρες θα ραβόντουσαν πάνω του ενώ στη δεύτερη περίπτωση θα τους κολλούσαμε με κάποιο είδος κόλλας. Επιλέξαμε την πρώτη περίπτωση, καθώς θεωρήσαμε ότι θα ήταν πιο εύκολο να μετακινήσουμε και να ξανατοποθετήσουμε τους αισθητήρες σε διαφορετική θέση, εάν πιστεύαμε ότι θα είχαμε καλυτέρα αποτελέσματα. Αντίθετα, στο πλαστικό γάντι αυτό δεν θα αποτελούσε επιλογή, καθώς θα ήταν σχεδόν αδύνατον να τους ξεκολλήσουμε χωρίς να υπάρχει κίνδυνος να χαλάσουν οι αισθητήρες. Επίσης, ακόμη ένας λόγος που μας ώθησε να διαλέξουμε το υφασμάτινο γάντι ήταν ότι είναι αισθητικά πιο ωραίο από το πλαστικό. Το μόνο αρνητικό είναι ότι σε παρατεταμένη χρήση, επειδή το γάντι είναι σχεδιασμένο για χειμώνα, μπορεί να υπάρξει ενόχληση στο άτομο που το φοράει.



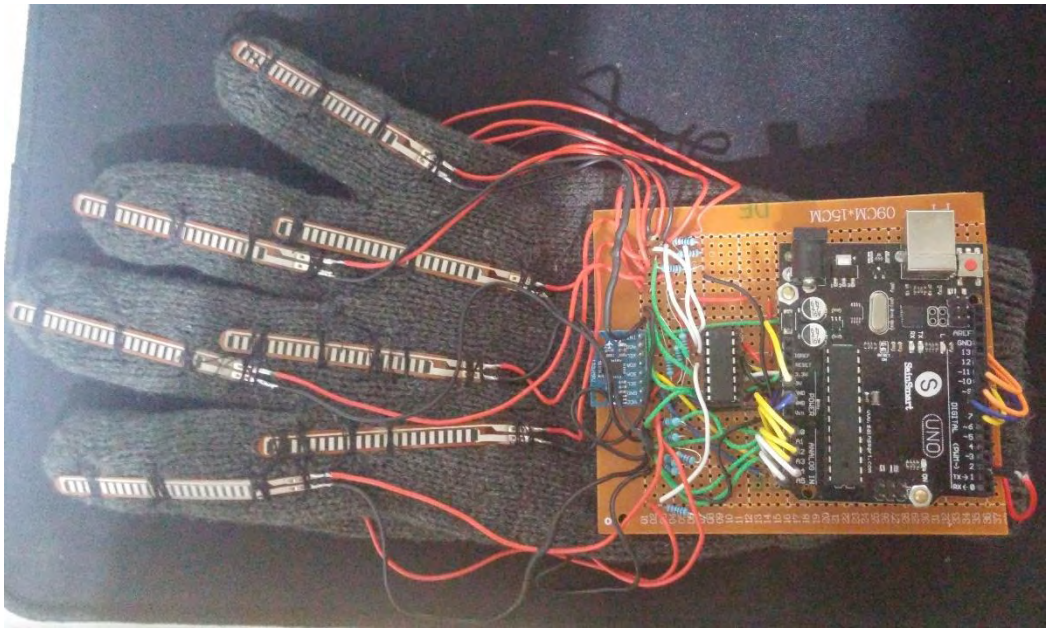
Εικόνα 25. Οι αισθητήρες πάνω στο γάντι

Το γάντι παρέχει αρκετή ευλυγισία στα δάχτυλα, χωρίς να υπάρχει κάποια ενόχληση σε οποιαδήποτε κίνηση θέλει να κάνει το άτομο που το φοράει. Επίσης, έτσι όπως είναι ραμμένοι οι αισθητήρες, παίρνουν τη μορφή του δαχτύλου και λυγίζουν με τον ίδιο τρόπο που λυγίζουν τα δάχτυλα.

3.2.4 Τελική μορφή

Τέλος, έγιναν οι συνδέσεις μεταξύ των αισθητήρων και της πλακέτας (protoboard) και προστέθηκαν όλοι οι απαραίτητοι αντιστάτες που χρειαζόντουσαν για λειτουργήσουν οι αισθητήρες με το Arduino. Αφού ολοκληρώθηκε η συσκευή, ήμασταν έτοιμοι να περάσουμε στην επόμενη φάση η οποία περιλάμβανε την ανάπτυξη του αλγορίθμου και τις πρώτες δοκιμές αναγνώρισης των γραμμάτων του ελληνικού νοηματικού αλφάβητου.

Για όλες τις συνδέσεις μεταξύ των αισθητήρων και της πλακέτας (protoboard), όπως επίσης και για όλες τις συνδέσεις μεταξύ των διάφορων κομματιών πάνω στην πλακέτα (protoboard) (ADC, resistors), χρησιμοποιήσαμε 22-23 AWG καλώδιο. Επίσης, για τις συνδέσεις μεταξύ του Arduino και της πλακέτας (protoboard) χρησιμοποιήσαμε jumper wires, τα οποία κόψαμε για να μικρύνει το μέγεθός τους.



Εικόνα 26. Τελική μορφή συσκευής

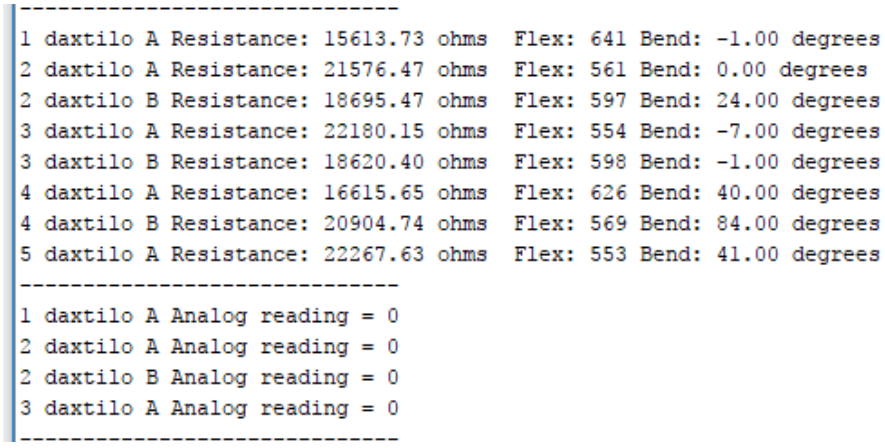
3.3 Ανάπτυξη Λογισμικού

Για να λειτουργήσει η συσκευή χρειαζόταν ένας αλγόριθμος/πρόγραμμα που θα διάβαζε τις τιμές των αισθητήρων κλίσης και επαφής, καθώς και του γυροσκοπίου, και θα αναγνώριζε το σωστό γράμμα του αλφάβητου.

3.3.1 Βασική μορφή του προγράμματος

Ο τρόπος που σκεφτήκαμε ήταν ο εξής: αφού το πρόγραμμα διαβάσει τις τιμές από τους αισθητήρες θα συγκρίνει αυτές τις τιμές με μια βάση δεδομένων που θα έχει αποθηκευμένες αυτές τις τιμές από προηγούμενες δοκιμές. Το πρόγραμμα που αποφασίσαμε να χρησιμοποιήσουμε ήταν το Arduino Ide το οποίο λειτουργεί άψογα με το Arduino Uno Board, που είχαμε ήδη αποφασίσει να χρησιμοποιήσουμε. Έγιναν αρκετές προσπάθειες και δοκιμές μέχρι να φτάσουμε στην τελευταία και πιο αποδοτική έκδοση του κώδικα.

Στην αρχή απλώς διαβάζαμε τις τιμές από τους αισθητήρες κλίσης και επαφής και τους εμφανίζαμε στην οθόνη (Εικόνα 27). Στη συνέχεια πήραμε ένα από τα παραδείγματα κώδικα σχετικά με τον τρόπο που λειτουργεί το γυροσκόπιο που κατεβάσαμε από τη βιβλιοθήκη (Rowberg, 2013) και αρχίσαμε να κάνουμε μετατροπές πάνω σε αυτό για να το ενσωματώσουμε στον δικό μας κώδικα.



Εικόνα 27. Οι τιμές που επιστρέφουν οι αισθητήρες

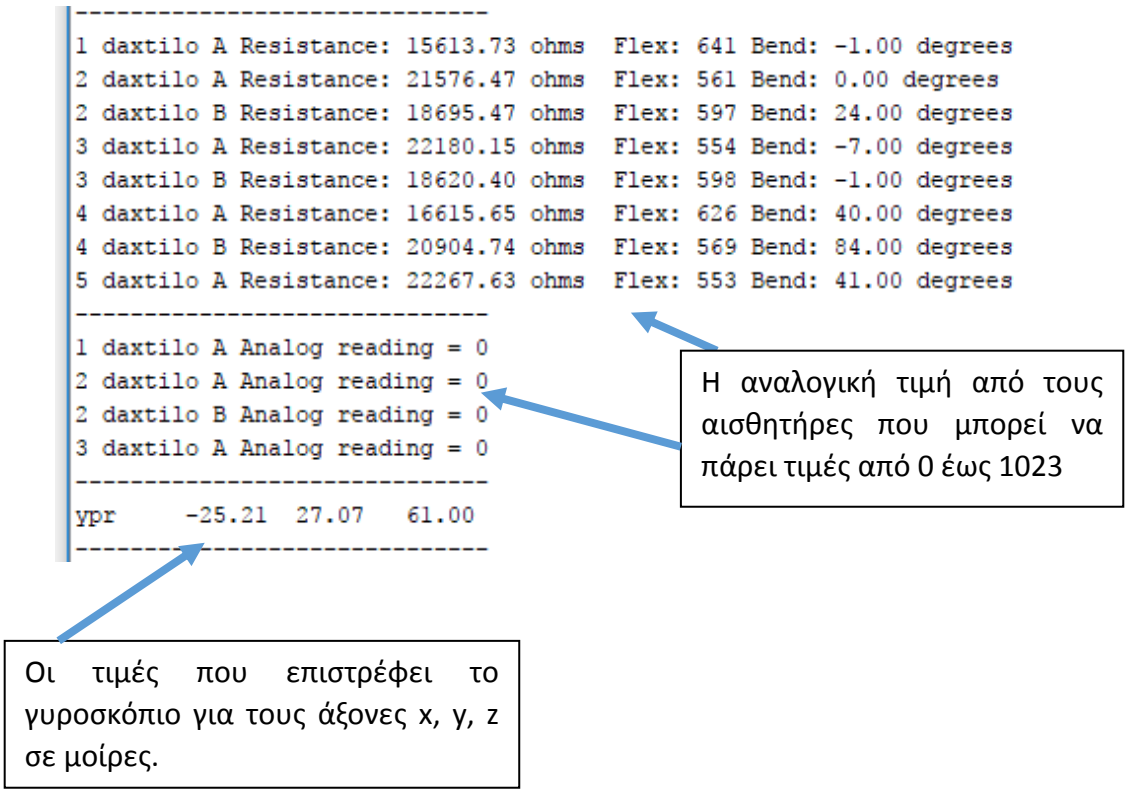
Πήραμε όλο τον αναγκαίο κώδικα από το παράδειγμα και τον χωρίσαμε σε δύο κομμάτια. Το πρώτο κομμάτι ονομάζεται setupGyro() και περιλαμβάνει όλες τις απαραίτητες διαδικασίες για να ετοιμαστεί το γυροσκόπιο και να αρχίσει να επιστρέφει τις τιμές των τριών αξόνων (x, y, z). Το δεύτερο κομμάτι ονομάζεται getGyroValues() και ενημερώνει τον global πίνακα ypr με τις καινούριες τιμές των αξόνων x, y, z κάθε φορά που καλείται η συνάρτηση. Σε αυτό το σημείο αξίζει να αναφερθεί ότι στις ρυθμίσεις που γίνονται στο γυροσκόπιο στη συνάρτηση setupGyro() υπάρχουν τέσσερις μεταβλητές οι οποίες διαφέρουν με κάθε συσκευή. Δηλαδή, είναι ένα είδος calibration για το κάθε γυροσκόπιο. Για να υπολογιστεί η κάθε μεταβλητή και η τιμή της βάσης του δικού μας γυροσκοπίου «τρέξαμε» ένα πρόγραμμα (Rowberg, 2013). Το πρόγραμμα μας εμφάνισε τέσσερις τιμές και εμείς τις χρησιμοποιήσαμε στην συνάρτηση setupGyro, ώστε να μας επιστρέφονται σωστά αποτελέσματα για τους άξονες x, y, z (Εικόνα 28).

```
// supply your own gyro offsets here, scaled for min sensitivity
mpu.setXGyroOffset(73);
mpu.setYGyroOffset(19);
mpu.setZGyroOffset(4);
mpu.setZAccelOffset(1000); // 1688 factory default for my test chip
```

Εικόνα 28. Συναρτήσεις για την παραμετροποίηση του γυροσκοπίου

Αφού ρυθμίσαμε το γυροσκόπιο, δημιουργήσαμε ένα πρόγραμμα που διάβαζε όλες τις τιμές από όλους τους αισθητήρες και τους εκτύπωνε στην οθόνη (

***Εικόνα 29*).**



Εικόνα 29. Οι τιμές των αισθητήρων μαζί με το γυροσκόπιο

Αφού λοιπόν καταφέραμε να διαβάζουμε όλες τις τιμές από τους αισθητήρες, συνεχίσαμε με την ανάπτυξη του αλγορίθμου για την αναγνώριση των γραμμάτων. Χωρίσαμε το κάθε είδος του αισθητήρα στη δική του συνάρτηση (αισθητήρες κλίσης: getFlexReding(), αισθητήρες επαφής: getFsrReading(), γυροσκόπιο: getGyroValues()) και στον δικό του πίνακα με τιμές (αισθητήρες κλίσης: unsigned int flexReading[8], αισθητήρες επαφής: int unsigned fsrReading[4], γυροσκόπιο: float ypr[3]), για να έχουμε καλύτερο έλεγχο και για να μπορούμε να ξεχωρίσουμε τον

κώδικα πιο εύκολα. Η κάθε συνάρτηση ενημερώνει τον δικό της πίνακα που περιέχει όλες τις τιμές των αισθητήρων. Επίσης, δημιουργήσαμε και έναν πίνακα (const int16_t savedLetters[][14]) που θα έχει αποθηκευμένες όλες τις τιμές των γραμμάτων για όλους τους αισθητήρες. Στην αρχή αποφασίσαμε να δοκιμάσουμε μόνο τρία γράμματα από το αλφάβητο για να δοκιμάσουμε κατά πόσο λειτουργεί ο αλγόριθμος που αναπτύξαμε.

3.3.2 Πρώτος αλγόριθμος

Ο πρώτος αλγόριθμος που σκεφτήκαμε γρήγορα απορρίφθηκε γιατί αποφασίσαμε ότι δεν θα ήταν πρακτικός για τη συγκεκριμένη εφαρμογή. Ο αλγόριθμος θα έπαιρνε τις τιμές από τους αισθητήρες και θα έλεγχε αν αυτές οι τιμές είναι μεταξύ κάποιων προκαθορισμένων τιμών για το κάθε γράμμα με ένα περιθώριο λάθους 5-10%. Δηλαδή, αν για το γράμμα «Α» η προκαθορισμένη τιμή για τον αισθητήρα κλίσης σε ένα από τα δάχτυλα είναι 121, θα έπρεπε η λαμβανόμενη τιμή από τον αισθητήρα την ώρα της δοκιμής να είναι μεταξύ 109 και 133. Όμως, επειδή όλοι οι αισθητήρες, με εξαίρεση αυτόν του γυροσκοπίου, επιστρέφουν διαφορετικές τιμές ανάλογα με το πώς έχει φορεθεί το γάντι ή με το αν η κίνηση γίνεται ακριβώς με τον ίδιο τρόπο που έγινε την πρώτη φορά, αναγκαστήκαμε να αλλάξουμε τον αλγόριθμο. Μια λύση που σκεφτήκαμε ήταν να αυξήσουμε το ποσοστό λάθους και να αφήσουμε μεγαλύτερο εύρος μεταξύ της προκαθορισμένης τιμής και της τιμής την ώρα της δοκιμής. Σε αυτή την περίπτωση όμως παρατηρήσαμε ότι το ποσοστό λάθους αναγνώρισης αυξήθηκε αρκετά, σε επίπεδο που η συσκευή πλέον θα ήταν αναξιόπιστη.

3.3.3 Δεύτερος αλγόριθμος

Αποφασίστηκε, επομένως, να αλλάξουμε τον αλγόριθμο και ο επόμενος και τελικός αλγόριθμος παίρνει τις τιμές από τους αισθητήρες και ψάχνει να βρει το γράμμα με τη μικρότερη διαφορά. Δηλαδή, παίρνει μία-μία τιμή από τον κάθε αισθητήρα και υπολογίζει την απόλυτη διαφορά με την αντίστοιχη αποθηκευμένη τιμή για το κάθε γράμμα και την αποθηκεύει. Έπειτα, συνεχίζει στον επόμενο αισθητήρα και προσθέτει την καινούρια διαφορά στην προηγούμενη ενώ η διαδικασία συνεχίζεται για όλους τους αισθητήρες. Είναι σημαντικό να αναφέρουμε σε αυτό το σημείο ότι

η τιμή του άξονα x από το γυροσκόπιο δεν λαμβάνεται υπ' όψιν διότι σε αυτή τη φάση δεν χρειαζόταν, καθώς τα αποτελέσματα που μας δίνουν οι αισθητήρες είναι αρκετά για να κάνουμε τη σύγκριση. Έτσι, δεν λαμβάνουμε υπ' όψιν τον άξονα που μας δείχνει την κατεύθυνση. Αφού βρούμε όλες τις διαφορές μεταξύ των τιμών για όλα τα γράμματα βρίσκουμε τη μικρότερη διαφορά και με αυτόν τον τρόπο αναγνωρίζεται το γράμμα.

Στις πρώτες εκδόσεις του δεύτερου αλγορίθμου, και αφού αρχίσαμε να προσθέτουμε γράμματα, χρησιμοποιούσαμε ως τιμές σύγκρισης τις μοίρες κλίσης του αισθητήρα κλίσης. Για να υπολογίσουμε τις μοίρες του κάθε αισθητήρα κλίσης, παρέχουμε στο πρόγραμμα δύο τιμές για τον καθένα που αποθηκεύονται σε δύο πίνακες. Στον πρώτο πίνακα αποθηκεύεται η αντίσταση του αισθητήρα στις 0^{o} μοίρες (straightResistance[]) ενώ στον δεύτερο πίνακα αποθηκεύεται η αντίσταση του αισθητήρα στις 90^{o} (bendResistance[]). Παρατηρήθηκε όμως ότι οι τιμές αυτές αλλάζουν κάθε φορά που ο χρήστης φοράει το γάντι, για αυτόν τον λόγο δημιουργήσαμε μια συνάρτηση (calibrate).

3.3.4 Συνάρτηση calibrate

Η συνάρτηση calibrate ζητάει από τον χρήστη να κρατήσει σταθερό το χέρι του στις 0^{o} μοίρες και, όταν σταθεροποιηθούν οι τιμές που λαμβάνονται από τον αισθητήρα, τις θέτει στον πίνακα straightResistance. Στη συνέχεια ζητάει από τον χρήστη να κρατήσει σταθερό το χέρι του στις 90^{o} μοίρες και με την ίδια λογική, όπως πριν, αποθηκεύει αυτές τις τιμές στον πίνακα bendResistance. Έτσι, κάθε φορά που ενεργοποιείται η συσκευή οι μοίρες που εμφανίζονται αντιστοιχούν στις πραγματικές.

3.3.5 Τελική έκδοση αλγορίθμου

Στην επόμενη και τελική έκδοση αποφασίσαμε για άλλη μια φορά να αλλάξουμε τον αλγόριθμο, διότι, όταν αρχίσαμε να προσθέτουμε περισσότερα γράμματα στις δοκιμές μας, ο αλγόριθμος άρχισε να κάνει λάθη καθώς δεν υπήρχαν μεγάλες αλλαγές μεταξύ των μοιρών του κάθε γράμματος του αλφάβητου. Οπότε, σκεφτήκαμε είτε να χρησιμοποιήσουμε την τιμή της αντίστασης που αλλάζει αρκετά, διότι όπως είπαμε και στο Κεφάλαιο 2 ένας αισθητήρας κλίσης μπορεί να

πάρει ένα μεγάλος εύρος τιμών kΩ, είτε να χρησιμοποιήσουμε την αναλογική τιμή που λαμβάνεται από το Arduino Uno. Επιλέξαμε τον δεύτερο τρόπο, διότι πιστέψαμε ότι δεν χρειαζόμαστε τόσο μεγάλος εύρος και θα είναι πιο εύκολο και πιο πρακτικό να χρησιμοποιήσουμε τις αναλογικές τιμές που έχουν εύρος από 0 έως 1023. Πράγματι, όπως θα δούμε και στο επόμενο κεφάλαιο με τα αποτελέσματα, οι αναλογικές τιμές είναι αρκετές για να κάνουμε την σύγκριση.

Με βάση τη σύγκριση των αισθητήρων επαφής σε όλους τους αλγορίθμους θεωρήσαμε ότι, εάν η τιμή του αισθητήρα είναι μεγαλύτερη του 10, τότε υπάρχει επαφή μεταξύ των δαχτύλων. Διαλέξαμε την τιμή 10, και όχι μικρότερη, διότι παρατηρήσαμε ότι μπορεί με μικρή επαφή, είτε κατά λάθος είτε επειδή η συγκεκριμένη χειρομορφή είναι περίπλοκη, να έχουμε τιμές μικρότερες του 10 που είναι λανθασμένες. Σε αυτούς τους αισθητήρες η σύγκριση γίνεται ως εξής:

- Εάν το συγκεκριμένο γράμμα έχει επαφή μεταξύ των δαχτύλων, όπως το γράμμα «Δ», τότε στον πίνακα υπάρχει η τιμή «1» για τον συγκεκριμένο αισθητήρα που υπάρχει επαφή ενώ για τους άλλους η τιμή «0». Η τιμή αυτή συγκρίνεται με την τιμή που λαμβάνεται από τον αισθητήρα, όταν προσπαθεί να γίνει η αναγνώριση, και, εάν είναι μεγαλύτερη του 10, τότε δεν προσθέτει καμία διαφορά στη συνολική διαφορά του γράμματος. Εάν όμως δεν είναι μεγαλύτερη του 10, τότε προσθέτει 100 μονάδες στη συνολική διαφορά του γράμματος.
- Εάν το γράμμα δεν έχει καμία επαφή μεταξύ των δαχτύλων, η τιμή που λαμβάνεται από τον αισθητήρα δεν προσθέτει καμία διαφορά. Ενώ, αν η τιμή του είναι μεγαλύτερη από 10, προσθέτει 100 μονάδες στη συνολική διαφορά του γράμματος.

Η σύγκριση των τιμών του γυροσκοπίου ακολουθεί ακριβώς την ίδια λογική με τους αισθητήρες κλίσης χωρίς όμως, όπως είπαμε, να λαμβάνεται υπ’ όψιν η τιμή του άξονα της κατεύθυνσης.

Όταν αρχίσαμε να προσθέτουμε περισσότερα γράμματα παρατηρήσαμε ότι θα είχαμε πρόβλημα με τη μνήμη του Arduino. Το Arduino παρέχει 8k bytes στο πρόγραμμα για αποθήκευση μεταβλητών. Ο πίνακάς μας για τα 24

γράμματα και τις 14 τιμές των αισθητήρων έχει 336 τιμές. Η κάθε τιμή, επειδή είναι int16_t, είναι 16 bits, δηλαδή 2 byte. Οπότε συνολικά θα χρειαζόντουσαν 672 bytes. Όμως το πρόγραμμα με τις απαραίτητες βιβλιοθήκες και άλλες μεταβλητές χρησιμοποιούσε ήδη 947 bytes. Παρατηρήσαμε ότι το Arduino Uno μετά από ένα ποσοστό γεμάτης μνήμης άρχισε να εμφανίζει απρόσμενες συμπεριφορές, όπως να αλλοιώνει τις τιμές των αισθητήρων ή να μην εκτελεί σωστά τον κώδικα. Οπότε αναγκαστήκαμε να πάρουμε κάποια μέτρα για να μειώσουμε το ποσοστό μνήμης που χρειάζεται το πρόγραμμα μας (Arduino, 2017):

- Ψάξαμε και βρήκαμε όλες τις μεταβλητές και κάναμε τις εξής αλλαγές:
    - Όποιες ήταν σταθερές και δεν άλλαζαν κατά τη διάρκεια εκτέλεσης του κώδικα τις κάναμε const
    - Μετατράπηκαν οι μεταβλητές οι οποίες ήταν float και μπορούσαν να γίνουν int.
- Αλλάξαμε όλες τις εκτυπώσεις μηνυμάτων και προσθέσαμε F(). Όταν ένα μήνυμα εκτυπώνεται, αντιγράφεται από τη μνήμη του προγράμματος (flash) στη SRAM και καταλαμβάνει χώρο. Αυτός ο τρόπος χρήσης δεν είναι σωστός διότι δεν πρόκειται να αλλάξει ποτέ το περιεχόμενο των μηνυμάτων και απλώς καταλαμβάνουν χώρο. Οπότε ο Paul Stoffregen δημιούργησε τη συνάρτηση F() σύμφωνα με την οποία ο μεταγλωττιστής (compiler) κρατάει τα μηνύματα (strings) στην PROGEM και δεν τα αντιγράφει στη SRAM.

Όμως, ακόμα και με τα παραπάνω βήματα, δεν είχαμε αρκετό χώρο να «τρέξουμε» το πρόγραμμα χωρίς να δημιουργούνται προβλήματα. Διαβάσαμε ωστόσο στην επίσημη σελίδα του Arduino (Arduino, 2017) ότι σε μια τέτοια περίπτωση μπορούμε να δοκιμάσουμε να χρησιμοποιήσουμε EEPROM μνήμη. Όπως είπαμε και στο Κεφάλαιο 3, η EPRROM δεν σβήνεται με τη διακοπή του ρεύματος και θεωρείται ως read-only memory, δηλαδή ο σκοπός της είναι κυρίως να διαβάζονται δεδομένα και να μην τροποποιούνται. Έτσι, με τη χρήση της EEPROM βιβλιοθήκης μπορέσαμε και αποθηκεύσαμε ολόκληρο τον πίνακα, με τις αποθηκευμένες τιμές των αισθητήρων

για τα γράμματα του αλφάβητου, στην EEPROM, με αποτέλεσμα να απελευθερώσουμε αρκετό χώρο στην SRAM για να «τρέχει» το πρόγραμμά μας χωρίς προβλήματα.

4. Αποτελέσματα

Αφού τελείωσε ο σχεδιασμός και η ανάπτυξη του αλγορίθμου, όπως και η κατασκευή της συσκευής, αποφασίσαμε ότι πρέπει να ελέγξουμε κατά πόσο καλύψαμε τις απαιτήσεις που θα μπορούσε να έχει μια τέτοια συσκευή (Άλλες απαιτήσεις).

4.1 Οικονομία

Για να αποφασίσουμε αν η συσκευή που κατασκευάσαμε θα είναι οικονομική, πρέπει να υπολογίσουμε το κόστος των υλικών που χρησιμοποιήσαμε. Συνολικά τα υλικά που χρησιμοποιήθηκαν στην συσκευή, χωρίς να υπολογίσουμε τα μεταφορικά, ανήλθε στα 135,57€. Πολλά όμως από τα υλικά, όπως οι αντιστάτες, αγοράστηκαν σε πακέτα, δηλαδή υπήρχαν αντιστάτες περισσότεροι από αυτούς που χρησιμοποιήθηκαν. Επίσης, η αγορά του breadboard, που υπολογίζουμε στην παραπάνω τιμή, χρησιμοποιήθηκε μόνο για δοκιμές που αφορούσαν τον σχεδιασμό της συσκευής και δεν θα είναι απαραίτητη σε περίπτωση μαζικής παραγωγής. Επιπλέον, θα μπορούσε να αντικατασταθεί το Arduino Uno με ένα barebone Arduino και να δημιουργηθεί μια PCB πλακέτα, δηλαδή μια πλακέτα που θα έχει μόνο τα απαραίτητα υλικά για λειτουργήσει η συσκευή. Αν υπολογίσουμε με βάση τις ενδεικτικές τιμές τη δημιουργία μιας τέτοιας συσκευής προκύπτει το εξής κόστος:

Υλικό	Ποσότητα	Τιμή	Συνολικά
Atmega chip	1	1,350 €	1,350 €
Κεραμικός πυκνωτής	2	0,060 €	0,120 €
Πυκνωτές	2	0,300 €	0,600 €
Clock Crystal	1	0,300 €	0,300 €
Voltage Regulator	1	0,100 €	0,100 €
Button	1	0,031 €	0,031 €
Flex Sensor	8	7,500 €	60,000 €
Gyroscope	1	1,510 €	1,510 €

Contact Sensor	4	6,530 €	26,120 €
ADC	1	3,840 €	3,840 €
PCB	1	1,420 €	1,420 €
Resistors	14	0,010 €	0,140 €
		Συνολικά	95,531€

Πίνακας 3. Κόστος συσκευής και βασικών περιφερειακών

Οπότε το κόστος, μόνο για τα απαραίτητα υλικά, είναι 96 Ευρώ. Βέβαια, σε αυτά πρέπει να συνυπολογίσουμε τις εργατοώρες, τον εξοπλισμό, τη μεταφορά καθώς και την τιμή χονδρικής. Όμως σε αυτή την περίπτωση οι τιμές όλων των υλικών θα είναι πολύ χαμηλότερες, αφού θα παράγονται μαζικά. Ενδεικτικά, οι αισθητήρες κλίσης και επαφής, αν κατασκευαστούν και δεν αγοραστούν, έχουν χαμηλότερο κόστος παραγωγής (κόστος υλικών) σε σχέση με την τιμή που αγοράστηκε.

4.2 Ευκολία στη χρήση

Οι παράγοντες που θα μπορούσαν να καθορίσουν την ευκολία στη χρήση είναι αφενός το κατά πόσο ένας χρήστης χωρίς εκτεταμένες γνώσεις πάνω στους ηλεκτρονικούς υπολογιστές μπορεί να χρησιμοποιήσει το γάντι, αφετέρου ο χρόνος που χρειάζεται προκειμένου να μπορεί κάθε φορά να το χρησιμοποιεί για να μεταφράζει τις χειρομορφές του.

Η συσκευή δεν απαιτεί κάποιες ιδιαίτερες γνώσεις για να χρησιμοποιηθεί, το μόνο που χρειάζεται από τον χρήστη είναι να φορέσει το γάντι και να το συνδέσει με τον ηλεκτρονικό υπολογιστή καθώς και να ανοίξει το πρόγραμμα (Arduino IDE). Μόλις γίνει αυτό, σε ελάχιστο χρόνο, μπορεί να ξεκινήσει να μεταφράζει χειρομορφές και να βλέπει τα αποτελέσματα στον υπολογιστή του. Παρ’ όλα αυτά, το γάντι σε αυτή τη φάση δεν έχει τη φορητότητα που θα μπορούσε να έχει. Για να λειτουργήσει, όπως είπαμε, απαιτεί φυσική σύνδεση (καλώδιο) με τον ηλεκτρονικό υπολογιστή.

4.3 Αξιοπιστία και ταχύτητα

Ο παράγοντας της αξιοπιστίας και της ταχύτητας της συσκευής είναι ίσως ο πιο σημαντικός παράγοντας από τις απαιτήσεις της συσκευής που πρέπει να ελεγχθεί σχετικά με το αν επιτεύχθηκε.

Αποφασίσαμε να δοκιμάσουμε όλα τα γράμματα του αλφάβητου, αφού ήδη είχαμε αποθηκεύσει τις τιμές των αισθητήρων του κάθε γράμματος και είχαμε αναπτύξει το πρόγραμμα. Η δοκιμή ξεκίνησε με το γράμμα «Α» και τελείωσε με το γράμμα «Ω». Η κάθε χειρομορφή του γράμματος ελεγχόταν τρεις φορές και η συνολική διαδικασία επαναλήφθηκε δύο φορές. Κάθε φορά που γινόταν έλεγχος του γράμματος αφήναμε χαλαρό το χέρι για ένα μικρό χρονικό διάστημα. Αφού ολοκληρώσαμε τον πρώτο έλεγχο, βγάλαμε το γάντι και στη συνέχεια το ξαναφορέσαμε, με σκοπό να δούμε εάν θα είχαμε διαφορά στα αποτελέσματα. Όταν φορούσαμε το γάντι δίναμε προσοχή ώστε τα δάχτυλα να είναι καλά τοποθετημένα στο γάντι και οι αισθητήρες του ευθυγραμμισμένοι με τα δάχτυλα. Στον παρακάτω πίνακα μπορείτε να δείτε τα αποτελέσματα για το κάθε γράμμα που δοκιμάσαμε:

Γράμμα	**1η δοκιμή**	**Ακρίβεια**	**2η δοκιμή**	**Ακρίβεια**	**Μ.Ο.**
Α	3	100%	3	100%	100,00%
Β	3	100%	3	100%	100,00%
Γ	3	100%	3	100%	100,00%
Δ	3	100%	3	100%	100,00%
Ε	1	33,3%	2	66,6%	50,00%
Ζ	3	100%	3	100%	100,00%
Η	3	100%	3	100%	100,00%
Θ	3	100%	3	100%	100,00%
Ι	3	100%	3	100%	100,00%
Κ	3	100%	3	100%	100,00%
Λ	2	66,6	2	66,6	66,60%
Μ	3	100%	3	100%	100,00%
Ν	3	100%	3	100%	100,00%
Ξ	3	100%	3	100%	100,00%
Ο	3	100%	3	100%	100,00%
Π	3	100%	3	100%	100,00%
Ρ	3	100%	3	100%	100,00%

Σ	0	0%	1	33,3%	16,60%
Τ	1	33,3%	1	33,3%	33,30%
Υ	3	100%	3	100%	100,00%
Φ	3	100%	3	100%	100,00%
Χ	3	100%	3	100%	100,00%
Ψ	3	100%	3	100%	100,00%
Ω	3	100%	2	66,6	83,30%
				Μ.Ο. ΣΥΝΟΛΙΚΑ	89.58%

Πίνακας 4. Ποσοστά σωστών προβλέψεων ανά γράμμα

Βάση του παραπάνω πίνακα η συσκευή μας σχεδόν στο 90% των περιπτώσεων αντιστοίχιζε σωστά τη χειρομορφή στο αντίστοιχο γράμμα του αλφάβητου. Στα αποτελέσματα μπορούμε να παρατηρήσουμε ότι μόνο σε πέντε γράμματα («Ε», «Λ», «Σ», «Τ», «Ω») είχαμε λάθος πρόβλεψη ενώ σε όλα τα υπόλοιπα είχαμε 100% σωστές προβλέψεις. Οι λάθος προβλέψεις για το γράμμα «Λ» οφείλονται στο γεγονός ότι η χειρομορφή του είναι πολύ κοντινή με το γράμμα «Ν». Στα υπόλοιπα τέσσερα γράμματα είναι πολύ πιθανό να είχαμε λάθος προβλέψεις είτε επειδή δεν ήμασταν σε θέση να αναπαραστήσουμε σωστά τη χειρομορφή, είτε επειδή κατά τη διάρκεια της πρώτης δειγματοληψίας δεν κάναμε σωστά τη χειρομορφή, είτε επειδή σημειώσαμε λάθος τα αποτελέσματα. Κατά τη διάρκεια των δοκιμών παρατηρήσαμε ότι οι τιμές που επιστρέφονται από το γυροσκόπιο είναι πολύ κοντινές σε αυτές που έχουμε αποθηκεύσει για το κάθε γράμμα και ότι η αποτυχία αναγνώρισης των γραμμάτων αυτών οφείλεται κυρίως στις τιμές που προέρχονται από τους αισθητήρες κλίσης. Το πιο πιθανό είναι την πρώτη φορά που σημειώθηκαν οι τιμές τα υπόλοιπα τα δάχτυλα να ήταν πιο «σφιχτά» καθώς αυτά τα πέντε γράμματα έχουν «περίπλοκες» χειρομορφές. Ενώ στις δοκιμές, επειδή το χέρι είχε εξασκηθεί περισσότερο σε όλες τις χειρομορφές, ήταν πιο «χαλαρό» με αποτέλεσμα να έχουμε λάθος προβλέψεις.

Για τον έλεγχο της ταχύτητας της συσκευής, εκτελέσαμε δύο δοκιμές. Στην πρώτη δοκιμή ελέγξαμε τον χρόνο που χρειάζεται για να ολοκληρωθούν όλες οι αρχικές ρυθμίσεις (setup) της συσκευής. Για να τον μετρήσουμε χρησιμοποιήσαμε μία μεταβλητή. Στη μεταβλητή «CurrentTime», μόλις τελείωσαν οι αρχικές ρυθμίσεις (setup), αποθηκεύσαμε τον συνολικό χρόνο που έχει περάσει από τη

στιγμή που άνοιξε η συσκευή σε μιλισεκόντ (Εικόνα 30). Η μέτρηση του χρόνου γίνεται με τη βοήθεια της συνάρτησης «millis()».

```
void setup()
{

  Serial.begin(115200);
  Serial.println(F("Start"));
  pinMode(CS_PIN, OUTPUT);
  pinMode(A0, INPUT);
  pinMode(A1, INPUT);
  pinMode(A2, INPUT);
  pinMode(A3, INPUT);
  calibrate();
  setupGyro();
  CurrentTime = millis();
  Serial.println(F("Stop"));
  Serial.println(CurrentTime);

}
```

Εικόνα 30. Παράδειγμα κώδικα μέτρησης ταχύτητας

Συνολικά «τρέξαμε» πέντε φορές την ίδια δοκιμή. Τα αποτελέσματα μπορούμε να τα δούμε στον παρακάτω πίνακα:

Δοκιμή	Χρόνος (ms)	Χρόνος (s)	Χρόνος χωρίς καθυστέρηση (delay) (s)
1	10118	10,118	4,118
2	10108	10,108	4,108
3	10117	10,117	4,117
4	10114	10,114	4,114
5	10116	10,116	4,116
		Μέσος όρος	4,1146

Πίνακας 5. Χρόνος που χρειάζεται για να ξεκινήσει η συσκευή

Είναι σημαντικό να αναφέρουμε εδώ ότι κατά τη διάρκεια των αρχικών ρυθμίσεων έχουμε προσθέσει έξι καθυστερήσεις, η κάθε μία από 1000 ms, για να δώσουμε στον χρήστη αρκετό χρόνο ώστε να προετοιμαστεί για αυτό που του ζητείται (Συνάρτηση calibrate). Οπότε, για να προετοιμαστεί η συσκευή και να είναι έτοιμη ώστε να μεταφράσει χειρομορφές χρειάζεται κατά μέσο όρο 4,1 δευτερόλεπτα. Είναι σημαντικό να μην υπάρχουν μεγάλες καθυστερήσεις σε αυτό το στάδιο, έτσι

ώστε να μην δημιουργείται δυσφορία στον χρήστη και να μπορεί να μεταφράσει χειρομορφές σχεδόν αμέσως αφού φορέσει το γάντι.

Στη δεύτερη δοκιμή ελέγξαμε τον χρόνο που χρειάζεται η συσκευή για να διαβάσει τα δεδομένα από τους διάφορους αισθητήρες και να ελέγξει ποια χειρομορφή ταιριάζει περισσότερο. Για να μετρήσουμε την ταχύτητα ακολουθήσαμε παρόμοια τεχνική με την προηγουμένη δοκιμή. Η μία διαφορά ήταν ότι δημιουργήσαμε μια δομή επανάληψης (loop) και «τρέξαμε» τέσσερις συναρτήσεις (Εικόνα 31). Η τρεις συναρτήσεις ήταν υπεύθυνες για να διαβάσουν τα δεδομένα από τους αισθητήρες μας και η τέταρτη συνάρτηση ήταν υπεύθυνη για να μεταφράσει τη χειρομορφή. Η δομή επανάληψης «έτρεχε» για 1000 φορές ενώ συνολικά τη δοκιμή την κάναμε πέντε φορές.

```
void loop()
{
  Serial.println(F("Start"));
   StartTime = millis();
 for(int i=0; i<1000; i++){


  getFlexReading();
  getFsrReading();
  getGyroValues();
    findLetter();

 }
   CurrentTime = millis();
ElapsedTime = CurrentTime - StartTime;
Serial.println(F("Stop"));
Serial.println(ElapsedTime);
```

Εικόνα 31. Παράδειγμα κώδικα μέτρησης ταχύτητας για τον αλγόριθμο

Για να μετρήσουμε τον χρόνο χρειαζόμασταν άλλες δύο μεταβλητές («StartTime», «ElapsedTime»). Τα αποτελέσματα φαίνονται στον παρακάτω πίνακα:

Δοκιμή	Χρόνος (ms)	Χρόνος (s)
1	8412	8,412
2	8409	8,409
3	8416	8,416
4	8414	8,414
5	8415	8,415
	Μέσος όρος	8,4132

Πίνακας 6. Χρόνος που χρειάζεται για την αναγνώριση της χειρομορφής

Επομένως, για μία επανάληψη χρειαζόντουσαν κατά μέσο όρο 0,008 δευτερόλεπτα. Είναι σημαντικό να διαβάζονται γρήγορα οι τιμές από τους αισθητήρες και να αναγνωρίζεται η χειρομορφή, διότι ένας νοηματιστής εκτελεί πολλές χειρομορφές σε πολύ σύντομο χρονικό διάστημα.

Από τα αποτελέσματα του χρόνου φτάνουμε στο συμπέρασμα ότι η συσκευή μας είναι αρκετά γρήγορη και στις δύο περιπτώσεις. Δεν καθυστερεί να ξεκινήσει, είναι λειτουργική και ο χρόνος που χρειάζεται για να αναγνωρίσει το γράμμα του αλφάβητου είναι αρκετά χαμηλός.

4.4 Αισθητική

Ένα προϊόν για να είναι επιτυχές στην αγορά θα πρέπει να είναι και αισθητικά ευχάριστο στον καταναλωτή. Θα πρέπει να μοιάζει με ένα κανονικό γάντι το οποίο δεν θα προκαλεί δυσφορία κατά τη χρήση του και ιδανικά θα μπορούσε να χρησιμοποιηθεί και κατά τη διάρκεια άλλων καθημερινών δραστηριοτήτων. Έτσι, δεν θα απαιτείται από τον καταναλωτή να το βγάζει και να το βάζει μόνο όταν θέλει να το χρησιμοποιήσει για τη μετάφραση χειρομορφών.

Από τις δικές μας παρατηρήσεις το γάντι φοριέται εύκολα και δεν προκαλεί δυσφορία. Το γάντι είναι όπως ένα φυσιολογικό γάντι που φοράμε τον χειμώνα. Επειδή όμως το γάντι μας έχει καλώδια, αισθητήρες κ.λπ., για να μην νιώθει ο χρήστης άβολα όταν το φοράει, θα μπορούσαμε να χρησιμοποιήσουμε ένα πιο λεπτό υφασμάτινο ή πλαστικό γάντι, έτσι ώστε να καλυφθούν τα διάφορα εξαρτήματα και να μην είναι ορατά από άλλους.

5. Συμπεράσματα και Μελλοντικές Επεκτάσεις

Καταφέραμε να δημιουργήσουμε μια συσκευή αναγνώρισης ελληνικών γραμμάτων του ελληνικού νοηματικού αλφαβήτου, η οποία σε ποσοστό σχεδόν 90% μπορεί να αναγνωρίσει επιτυχώς τη χειρομορφή σε πολύ μικρό χρόνο. Παρ' όλα αυτά, μπορούν να γίνουν βελτιώσεις.

Μια σημαντική βελτίωση θα ήταν να αυξηθεί το ποσοστό των σωστών προβλέψεων. Για κάθε χειρομορφή χρησιμοποιούμε μόνο ένα σετ από τιμές που προέρχονται από μία δοκιμή. Εάν χρησιμοποιούσαμε περισσότερες τιμές πιθανόν να είχαμε αύξηση αυτού του ποσοστού. Επίσης, δεν χρησιμοποιούμε τον άξονα της κατεύθυνσης που μας επιστρέφεται από το γυροσκόπιο και, όπως είπαμε σε προηγούμενα κεφάλαια, η κατεύθυνση του χεριού στις νοηματικές γλώσσες παίζει μεγάλο ρόλο στη μετάφραση των χειρομορφών.

Τέλος, πολλά μπορούν να γίνουν στην αισθητική του γαντιού καθώς και στη φορητότητά του. Το πιο βασικό είναι η αλλαγή της πλακέτας (protoboard) σε PCB, έτσι ώστε να μικρύνει το γάντι σε μέγεθος και πάχος. Ακόμα, σημαντικό θα ήταν να καλυφθούν όλα τα ηλεκτρονικά κομμάτια της συσκευής για να είναι πιο προσιτό στον χρήστη. Τέλος, η συσκευή θα μπορούσε να λειτουργήσει με μπαταρίες και η επικοινωνία να γίνει ασύρματη κάνοντας τη συσκευή φορητή.

Βιβλιογραφία

Ada, L. (2015, Νοέμβριος 19). *adafruit*. Ανάκτηση Ιούνιος 22, 2017, από Using an FSR: https://learn.adafruit.com/force-sensitive-resistor-fsr/using-an-fsr

Arduino. (2017). *Arduino*. Ανάκτηση Ιούνιος 18, 2017, από Arduino Memory: https://www.arduino.cc/en/Tutorial/Memory

Arduino. (2017). *Arduino*. Ανάκτηση Απρίλιος 28, 2017, από ARDUINO UNO REV3: https://store.arduino.cc/arduino-uno-rev3

Arduino. (2017). *Arduino*. Ανάκτηση Ιούνιος 22, 2017, από MPU-6050 Accelerometer + Gyro: https://playground.arduino.cc/Main/MPU-6050

Arduino. (2017). *Arduino*. Ανάκτηση Ιούνιος 22, 2017, από Arduino Memory: https://www.arduino.cc/en/Tutorial/Memory

Atmel. (2015, Δεκέμβριος 10). *Atmel*. Ανάκτηση Ιούνιος 13, 2017, από http://www.atmel.com/images/atmel-8271-8-bit-avr-microcontroller-atmega48a-48pa-88a-88pa-168a-168pa-328-328p_datasheet_complete.pdf

Dicola, T. (2016, Σεπτέμβριος 09). *Adafruit*. Ανάκτηση Απρίλιος 14, 2017, από MCP3008: https://learn.adafruit.com/raspberry-pi-analog-to-digital-converters/mcp3008

Helderman, A., Mehilli, R., & Cloutier, P. (2016, Απρίλιος 27). *wpi*. Ανάκτηση από Affordable Sign Language Glove: https://web.wpi.edu/Pubs/E-project/Available/E-project-042816-155206/unrestricted/SignLanguageGloveMQP.pdf

Hienzsch, D. (2015, Απρίλιος 10). *Rheingold Heavy*. Ανάκτηση Ιούνιος 10, 2017, από https://rheingoldheavy.com/spi-basics/

Hienzsch, D. (2015, Απρίλιος 20). *Rheingold Heavy*. Ανάκτηση Ιούνιος 12, 2017, από MCP3008 Tutorial 01: Functionality Overview: https://rheingoldheavy.com/mcp3008-tutorial-01-functionality-overview/

Interlink Electronics. (2017). *trossenrobotics*. Ανάκτηση Ιούνιος 19, 2017, από FSR 402 Data Sheet: http://www.trossenrobotics.com/productdocs/2010-10-26-DataSheet-FSR402-Layout2.pdf

Microchip. (2008, Φεβρουαριος 2). *Adafruit*. Ανάκτηση Ιούνιος 12, 2017, από MCP3004/3008: https://cdn-shop.adafruit.com/datasheets/MCP3008.pdf

pi-schools. (2017). *pi-schools*. Ανάκτηση Ιούνιος 18, 2017, από Η Ελληνική Νοηματική Γλώσσα: http://www.pi-schools.gr/special_education_new/html/gr/8emata/ekp_yliko/kofosi.htm

Rowberg, J. (2013, Αύγουστος 05). *github*. Ανάκτηση από I2C device library: https://github.com/jrowberg/i2cdevlib

Sanjeev, A. (2016). *diyhacking*. Ανάκτηση Μάιος 19, 2017, από diyhacking: https://diyhacking.com/arduino-mpu-6050-imu-sensor-tutorial/

Sparkfun. (2017). *Sparkfun*. Ανάκτηση Απρίλιος 19, 2017, από Flex Sensor Hookup Guide: https://learn.sparkfun.com/tutorials/flex-sensor-hookup-guide

Sparkfun. (2017). *Sparkfun*. Ανάκτηση Απρίλιος 18, 2017, από Flex Sensor 2.2": https://www.sparkfun.com/products/10264

Sparkfun. (2017). *Sparkfun*. Ανάκτηση Μάρτιος 4, 2017, από Flex Sensor 4.5": https://www.sparkfun.com/products/8606

Sparkfun. (2017). *Sparkfun*. Ανάκτηση Ιούνιος 8, 2017, από Force Sensitive Resistor 0.5": https://www.sparkfun.com/products/9375

Sparkfun. (2017). *SparkFun*. Ανάκτηση Ιούνιος 25, 2017, από Voltage Dividers.

SparkFun. (2017). *SparkFun*. Ανάκτηση Ιούνιος 25, 2017, από SparkFun Triple Axis Accelerometer and Gyro Breakout - MPU-6050: https://www.sparkfun.com/products/11028

SpectraSymbol. (2017). *sparkfun*. Ανάκτηση Ιούνιος 4, 2017, από sparkfun: https://www.sparkfun.com/datasheets/Sensors/Flex/FlexSensor.pdf

Spectrasymbol. (2017). *Spectrasymbol*. Ανάκτηση Μαιος 25, 2017, από FLEX SENSORS: http://www.spectrasymbol.com/product/flex-sensors/

Wikipedia. (2016, Ιούλιος 24). *Wikipedia*. Ανάκτηση Ιουνιος 15, 2017, από Force-sensing resistor: https://en.wikipedia.org/wiki/Force-sensing_resistor

Wikipedia. (2017, Ιούνιος 24). Ανάκτηση Ιούνιος 24, 2014, από Serial Peripheral Interface Bus: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

Wikipedia. (2017, Μάιος 11). *Wikipedia*. Ανάκτηση Ιούνιος 2, 2017, από Voltage divider: https://en.wikipedia.org/wiki/Voltage_divider

Wikipedia. (2017, Φεβρουαριος 17). *Wikipedia*. Ανάκτηση Ιουνιος 19, 2017, από Ελληνική νοηματική γλώσσα: https://el.wikipedia.org/wiki/%CE%95%CE%BB%CE%BB%CE%B7%CE%BD%CE%B9%CE%BA%CE%AE_%CE%BD%CE%BF%CE%B7%CE%BC%CE%B1%CF%84%CE%B9%CE%BA%CE%AE_%CE%B3%CE%BB%CF%8E%CF%83%CF%83%CE%B1

Invensense. (2013, Σεπτέμβριος 08). *invensense*. Ανάκτηση Ιούνιος 8, 2017, από invensense MPU-6050: https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

Καμαρινού, Σ., & Μεϊμάρη, Μ. (2016, Οκτώμβριος). *«Ανάπτυξη διαδικτυακής εφαρμογής δημιουργίας γλωσσικών.* Ανάκτηση Ιούνιος 25, 2017, από http://okeanis.lib.teipir.gr/xmlui/bitstream/handle/123456789/3182/cse39196.pdf?sequence=1

Καραπάντζος, Η. (2015). *Ανατομία του ανθρώπου.* Κυπρος: Broken Hill Publishers LTD.

Κέντρο Ελληνικής Νοηματικής Γλώσσας. (2017). Ανάκτηση Ιουνιος 14, 2017, από Κέντρο Ελληνικής Νοηματικής Γλώσσας: http://www.keng.gr/

Μπιτσάκης, Λ. (2015, Νοέμβριος). *Ανάπτυξη Android Εφαρμογής «Deaf – Blind Communicator».* Ανάκτηση Μαιος 25, 2017, από «Deaf – Blind Communicator»

Translatum Journal and the Author. (2002). *Translatum.* Ανάκτηση Ιούνιος 14, 2017, από Translatum Journal: Translatum Journal

Φραγκιουδάκη , Α., & Φραγκιουδάκη , Μ.-Ε. (2011, Σεπτεμβριος 12). *Νοηματική Γλώσσα: Οπτική Αναγνώριση και Δημιουργία-Σύνθεση.* Ανάκτηση από Οπτική Αναγνώριση και Δημιουργία-Σύνθεση

Παράρτημα Α

4.5 Libraries

```
#include <EEPROM.h>
#include <MCP3008.h>
long previousMillis = 0;
```

4.6 ADC

```
//---------------------------------------------------------------
//adc

//define pin connections
#define CS_PIN 12
#define CLOCK_PIN 9
#define MOSI_PIN 11
#define MISO_PIN 10

MCP3008 adc(CLOCK_PIN, MOSI_PIN, MISO_PIN, CS_PIN);
```

4.7 Flex Sensors

```
//---------------------------------------------------------------
//flex

unsigned int flexReading[8];
//1 Daxtilo A 2 daxtilo A 2 daxtilo B.......5 Daxtilo A
const String fingerNames[] = {"1 daxtilo A", "2 daxtilo A", "2
daxtilo B", "3 daxtilo A","3 daxtilo B", "4 daxtilo A","4 daxtilo B",
"5 daxtilo A"};
const unsigned  int flex_pins[] = {A3,A2,A1,A0,7,6,5,4};
unsigned int  straightResistance[] =
{19400,20400,15300,37300,37300,37300,37300,37300};
unsigned int bendResistance[] =
{30000,34500,27300,90000,90000,90000,90000,90000};

const  float VCC = 4.98; // Measured voltage of Ardunio 5V line
const  float R_DIV = 26200.0; // Measured resistance
```

4.8 Contact Sensors

```
//---------------------------------------------------------------
//fsr
int unsigned fsrReading[4];      // the analog reading from the FSR
resistor divider
```

```
//2 daxtilo 3 daxtilo ...5 daxtilo
const unsigned int fsr_pins[] = {3,2,1,0};
```

4.9 Gyroscope

```
//---------------------------------------------------------------
//gyro

#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif


MPU6050 mpu;

// MPU control/status vars
bool dmpReady = false;  // set true if DMP init was successful
uint8_t mpuIntStatus;   // holds actual interrupt status byte from
MPU
uint8_t devStatus;      // return status after each device operation
(0 = success, !0 = error)
uint16_t packetSize;    // expected DMP packet size (default is 42
bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars
Quaternion q;           // [w, x, y, z]         quaternion container
VectorFloat gravity;    // [x, y, z]            gravity vector
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll
container and gravity vector
```

4.10 Functions

```
//-----------------------------------------------
//functions
volatile bool mpuInterrupt = false;     // indicates whether MPU
interrupt pin has gone high
void getGyroValues();
void setupGyro();
void printData();
void getFlexReading();
void getFsrReading();
void calibrate();
void findLetter();
template <class T> int  EEPROM_readAnything(int ee, T& value);
unsigned long StartTime;
unsigned long CurrentTime;
unsigned long ElapsedTime;
```

4.11 setup()

```
//-------------------------------------------------
//setup

void setup()
{

    Serial.begin(115200);

    pinMode(CS_PIN, OUTPUT);
    pinMode(A0, INPUT);
    pinMode(A1, INPUT);
    pinMode(A2, INPUT);
    pinMode(A3, INPUT);

    //calibrate degrees
    calibrate();
    //initiate  gyro
    setupGyro();


}
```

4.12 Loop()

```
void loop()
{
    //get all sensor values
    getFlexReading();
    getFsrReading();
    getGyroValues();

    unsigned long currentMillis = millis();
    //print every 1.7 seconds
    //in order for the user to read the data
    if (currentMillis - previousMillis > 1700) {
        previousMillis = currentMillis;
        printData();

    }

}
```

4.13 Calibrate()

```
void calibrate() {

    //prepare the user
    Serial.println(F("Keep your hand still and straight"));
    Serial.print(F("In 3.."));
    delay(1000);
    Serial.print(F("2.."));
    delay(1000);
    Serial.print(F("1.."));
    delay(1000);
    Serial.println(F("Now"));
    getFlexReading();
    boolean flag = true;
```

```
float calibrate[8];
int count = 0;
//get the values from sensors
for (int i=0; i<8; i++) {
    calibrate[i] = flexReading[i];
}
//get at least 1000 samples and wait for the to be stable <40
while (flag || count < 1000) {
    getFlexReading();
    for (int i=0; i<8; i++) {
        if (abs(calibrate[i] - flexReading[i]) < 40) {
            flag = false;
        } else {
            flag = true;
        }
        calibrate[i] = flexReading[i];
    }
    count++;
}
Serial.print(F("Resistannce: "));
//calculate resistance and save it in the array
for (int i=0; i<8; i++) {
    float flexV = flexReading[i] * VCC / 1023.0;
    float flex = R_DIV * (VCC / flexV - 1.0);
    straightResistance[i] = flex;
    Serial.print(String(calibrate[i]) +", ");
}
//same thing for 90 degrees
Serial.println();
Serial.println(F("Keep your hand still and in 90 degrees"));
Serial.print(F("In 3.."));
delay(1000);
Serial.print(F("2.."));
delay(1000);
Serial.print(F("1.."));
delay(1000);
Serial.println(F("Now"));
getFlexReading();
flag = true;

count = 0;
for (int i=0; i<8; i++) {
    calibrate[i] = flexReading[i];
}

while (flag || count < 1000) {
    getFlexReading();
    for (int i=0; i<8; i++) {
        if (abs(calibrate[i] - flexReading[i]) < 40) {
            flag = false;
        } else {
            flag = true;
        }
        calibrate[i] = flexReading[i];
    }
    count++;
}
Serial.print(F("Resistannce: "));
for (int i=0; i<8; i++) {
    float flexV = flexReading[i] * VCC / 1023.0;
    float flex = R_DIV * (VCC / flexV - 1.0);
```

```
        bendResistance[i] = flex;
        Serial.print(String(calibrate[i]) +", ");
    }

    Serial.println(F("Calibration Done"));
}
```

4.14 getFlexReading()

```
void getFlexReading() {
    //read data from flex sensors
    //half of them are connected in the adc
    for (int i=0; i<8; i++) {
        int flexADC;
        if (i<4) {
            flexADC =analogRead(flex_pins[i]);
        } else {
            flexADC = adc.readADC(flex_pins[i]);


        }
        flexReading[i]= flexADC;

    }
}
```

4.15 getFsrReading()

```
void getFsrReading() {
    //get data from adc for the fsr sensors
    for (int i=0; i<4; i++) {
        fsrReading[i] = adc.readADC(fsr_pins[i]);

    }
}
```

4.16 findLetter()

```
void findLetter() {
    const int Letters = 24;
    String letter[24] = {"A" ,"B" , "G" ,"D","E","Z", "H" ,"8", "I",
"K" ,"L" , "M", "N" , "3" ,"O", "P" , "R", "S", "T", "Y", "F" ,"X" ,
"PS" , "W"};
//this array is saved now in the epprom memory
    /*
     const int16_t savedLetters[][14] = {
       {637,452,553,489,511,602,608,539,0,0,0,0,86,-3},
         {481,581,642,675,611,670,659,598,0,0,0,0,77,-2},
         {626,578,580,508,485,624,552,549,0,0,0,0,-73,5},
         {589,565,627,563,514,624,581,535,0,1,0,0,81,-2},
         {463,457,621,474,504,626,603,510,0,0,0,0,75,-3},
         {481,572,624,589,504,636,583,606,0,0,0,0,10,79},
         {475,573,609,462,504,600,596,602,0,0,0,0,82,-7},
         {488,588,634,582,606,555,635,548,0,0,0,0,15,71},
         {453,395,526,517,517,614,576,604,0,0,0,0,77,-12},
```

```
          {626,570,627,591,494,622,582,540,0,0,0,0,78,-4},
          {555,593,623,572,600,629,601,572,0,0,0,0,-64,13},
          {516,582,623,571,597,672,659,568,0,0,0,0,-72,14},
          {539,590,594,574,587,646,590,559,0,0,0,0,-72,5},
          {491,575,646,577,609,671,662,554,0,0,0,0,10,70},
          {507,465,537,516,512,611,611,527,0,0,0,0,64,-5},
          {528,580,629,501,485,626,593,612,0,0,0,0,-62,22},
          {554,597,564,567,616,586,613,558,0,1,0,0,77,-11},
          {587,467,544,494,590,610,582,521,1,1,0,0,86,-3},
          {522,585,633,438,517,588,620,593,0,0,0,0,-61,20},
          {620,462,550,583,492,626,568,604,0,0,0,0,80,-8},
          {641,520,527,586,601,668,662,606,1,0,0,0,82,-3},
          {451,528,614,523,604,567,602,550,0,0,0,0,82,-3},
          {530,579,584,584,561,665,644,610,0,0,0,1,83,-3},
          {632,547,554,589,527,653,613,598,0,1,1,0,49,8}

                                  };
                                  */

    int savedLetters[14];
    unsigned int memory = 0;

    int diff=0;

    int totaldiff;
    int minDiff = pow(2,15) -1 ;
    unsigned int finded;
    //loop for each letter
    for (int i=0; i<Letters; i++) {
        //get all 14 sensor values from epprom memory for each letter
        for (int w=0; w<14; w++) {
            EEPROM_readAnything(memory,savedLetters[w]);
            //Serial.print((String)savedLetters[w] +" ");
            //increase memory counter
            memory += 2;
        }
        //calculate difference for each type of sensor
        totaldiff = 0;
        diff=0;
        for (int j=0; j<8; j++) {
            diff = flexReading[j] - savedLetters[j];

            diff = abs(diff);
            totaldiff += diff;
            // Serial.println("Diff: " + String(diff));
        }
        for (int j=0; j<4; j++) {
            diff = (fsrReading[j] -
savedLetters[j+8]);//savedLetters[i][j+8]);
            diff = abs(diff);
            if (fsrReading[j] > 10) {
                if (savedLetters[j+8] == 8) {
diff =0;
                } else {
                    diff = 100;
                }
            } else {
                if (savedLetters[j+8] == 0) {
                    diff =0;
                } else {
                    diff = 100;
```

```
                }
            }

            totaldiff += diff;
        }
        for (int j=0; j<2; j++) {
            diff = ((ypr[j+1] * 180/M_PI) - savedLetters[j+12]);

            diff = abs(diff);
            totaldiff += diff;
        }

        Serial.println(("Total diff : " + String(totaldiff)) +
letter[i]);
        //update new min difference and keep the letter
        if ( totaldiff < minDiff) {

            minDiff = totaldiff;
            finded = i;

        } else if ( totaldiff  == minDiff) {

        }


    }
    //print the letter
    Serial.println("Finded index: " + letter[finded]+ " Diff : " +
String(minDiff) );
}
```

4.17 printData()

```
//-----------------------------------------------------
//print
void printData() {

    Serial.println(F("-----------------------------"));
    for (int i=0; i<8; i++) {
        int flexADC;
        float flexV = flexReading[i] * VCC / 1023.0;
        float flex = R_DIV * (VCC / flexV - 1.0);

        Serial.print(fingerNames[i] + " Resistance: " + String(flex)
+ " ohms  Flex: " + String(flexReading[i]) +" ");

        // Use the calculated resistance to estimate the sensor's
        // bend angle:
        float angle = map(flex, straightResistance[i],
bendResistance[i],
                          0, 90.0);
        Serial.println("Bend: " + String(angle) + " degrees  ");


    }
    Serial.println(F("-----------------------------"));


    for (int i=0; i<4; i++) {
```

```
        Serial.print(fingerNames[i] + " Analog reading = ");
        Serial.println(String(fsrReading[i]) +"    ");

    }
    Serial.println(F("-------------------------------"));

    Serial.print(F("ypr\t"));
    Serial.print(ypr[0] * 180/M_PI);
    Serial.print(F("\t"));
    Serial.print(ypr[1] * 180/M_PI);
    Serial.print(F("\t"));
    Serial.println(ypr[2] * 180/M_PI);
    Serial.println(F("-------------------------------"));

    findLetter();
}
```

4.18 gyro functions

```
//------------------------------------------------------
//gyro functions
```

4.18.1 dmpDataReady()

```
void dmpDataReady() {
    mpuInterrupt = true;
}
```

4.18.2 setupGyro()

```
void setupGyro() {
    // join I2C bus (I2Cdev library doesn't do this automatically)
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
    Wire.begin();
    TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)
#elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
    Fastwire::setup(400, true);
#endif




    // initialize device
    Serial.println(F("Initializing I2C devices..."));
    mpu.initialize();

    // verify connection
    Serial.println(F("Testing device connections..."));
    Serial.println(mpu.testConnection() ? F("MPU6050 connection
successful") : F("MPU6050 connection failed"));



    // load and configure the DMP
```

```
    Serial.println(F("Initializing DMP..."));
    devStatus = mpu.dmpInitialize();

    // supply your own gyro offsets here, scaled for min sensitivity
    mpu.setXGyroOffset(73);
    mpu.setYGyroOffset(19);
    mpu.setZGyroOffset(4);
    mpu.setZAccelOffset(1000); // 1688 factory default for my test
chip

    // make sure it worked (returns 0 if so)
    if (devStatus == 0) {
        // turn on the DMP, now that it's ready
        Serial.println(F("Enabling DMP..."));
        mpu.setDMPEnabled(true);

        // enable Arduino interrupt detection
        Serial.println(F("Enabling interrupt detection (Arduino
external interrupt 0)..."));
        attachInterrupt(0, dmpDataReady, RISING);
        mpuIntStatus = mpu.getIntStatus();

        // set our DMP Ready flag so the main loop() function knows
it's okay to use it
        Serial.println(F("DMP ready! Waiting for first
interrupt..."));
        dmpReady = true;

        // get expected DMP packet size for later comparison
        packetSize = mpu.dmpGetFIFOPacketSize();
    } else {
        // ERROR!
        // 1 = initial memory load failed
        // 2 = DMP configuration updates failed
        // (if it's going to break, usually the code will be 1)
        Serial.print(F("DMP Initialization failed (code "));
        Serial.print(devStatus);
        Serial.println(F(")"));
    }


}
```

4.18.3 getGyroValues()

```
void getGyroValues() {


    mpuIntStatus = mpu.getIntStatus();

    // get current FIFO count
    fifoCount = mpu.getFIFOCount();

    // check for overflow (this should never happen unless our code
is too inefficient)
    if ((mpuIntStatus & 0x10) || fifoCount == 1024) {
        // reset so we can continue cleanly
        mpu.resetFIFO();
        Serial.println(F("FIFO overflow!"));
```

```
        // otherwise, check for DMP data ready interrupt (this should
happen frequently)
    } else if (mpuIntStatus & 0x02) {
        // wait for correct available data length, should be a VERY
short wait
        while (fifoCount < packetSize) fifoCount =
mpu.getFIFOCount();

        // read a packet from FIFO
        mpu.getFIFOBytes(fifoBuffer, packetSize);

        // track FIFO count here in case there is > 1 packet
available
        // (this lets us immediately read more without waiting for an
interrupt)
        fifoCount -= packetSize;


        // display Euler angles in degrees
        mpu.dmpGetQuaternion(&q, fifoBuffer);
        mpu.dmpGetGravity(&gravity, &q);
        mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
        mpu.resetFIFO();



    }
}
```

4.19 EEPROM_readAnything

```
template <class T> int EEPROM_readAnything(int ee, T& value)
{
    byte* p = (byte*)(void*)&value;
    int i (0);
    for (; i < sizeof(value); ++i)
        *p++ = EEPROM.read(ee++);
    return i;
}
```

Παράρτημα Β

(Rowberg, 2013)

```
// I2Cdev and MPU6050 must be installed as libraries
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"

///////////////////////////////////   CONFIGURATION
/////////////////////////////
//Change this 3 variables if you want to fine tune the skecth to your
needs.
int buffersize=1000;     //Amount of readings used to average, make
it higher to get more precision but sketch will be slower
(default:1000)
int acel_deadzone=8;     //Acelerometer error allowed, make it lower
to get more precision, but sketch may not converge  (default:8)
int giro_deadzone=1;     //Giro error allowed, make it lower to get
more precision, but sketch may not converge  (default:1)

// default I2C address is 0x68
// specific I2C addresses may be passed as a parameter here
// AD0 low = 0x68 (default for InvenSense evaluation board)
// AD0 high = 0x69
//MPU6050 accelgyro;
MPU6050 accelgyro(0x68); // <-- use for AD0 high

int16_t ax, ay, az,gx, gy, gz;

int mean_ax,mean_ay,mean_az,mean_gx,mean_gy,mean_gz,state=0;
int ax_offset,ay_offset,az_offset,gx_offset,gy_offset,gz_offset;

///////////////////////////////////   SETUP
////////////////////////////////////
void setup() {
  // join I2C bus (I2Cdev library doesn't do this automatically)
  Wire.begin();
  // COMMENT NEXT LINE IF YOU ARE USING ARDUINO DUE
  TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz). Leonardo
measured 250kHz.

  // initialize serial communication
  Serial.begin(115200);

  // initialize device
  accelgyro.initialize();

  // wait for ready
  while (Serial.available() && Serial.read()); // empty buffer
  while (!Serial.available()){
    Serial.println(F("Send any character to start sketch.\n"));
    delay(1500);
  }
  while (Serial.available() && Serial.read()); // empty buffer again

  // start message
  Serial.println("\nMPU6050 Calibration Sketch");
  delay(2000);
```

```
  Serial.println("\nYour MPU6050 should be placed in horizontal
position, with package letters facing up. \nDon't touch it until you
see a finish message.\n");
  delay(3000);
  // verify connection
  Serial.println(accelgyro.testConnection() ? "MPU6050 connection
successful" : "MPU6050 connection failed");
  delay(1000);
  // reset offsets
  accelgyro.setXAccelOffset(0);
  accelgyro.setYAccelOffset(0);
  accelgyro.setZAccelOffset(0);
  accelgyro.setXGyroOffset(0);
  accelgyro.setYGyroOffset(0);
  accelgyro.setZGyroOffset(0);
}

////////////////////////////////////   LOOP
////////////////////////////////////
void loop() {
  if (state==0){
    Serial.println("\nReading sensors for first time...");
    meansensors();
    state++;
    delay(1000);
  }

  if (state==1) {
    Serial.println("\nCalculating offsets...");
    calibration();
    state++;
    delay(1000);
  }

  if (state==2) {
    meansensors();
    Serial.println("\nFINISHED!");
    Serial.print("\nSensor readings with offsets:\t");
    Serial.print(mean_ax);
    Serial.print("\t");
    Serial.print(mean_ay);
    Serial.print("\t");
    Serial.print(mean_az);
    Serial.print("\t");
    Serial.print(mean_gx);
    Serial.print("\t");
    Serial.print(mean_gy);
    Serial.print("\t");
    Serial.println(mean_gz);
    Serial.print("Your offsets:\t");
    Serial.print(ax_offset);
    Serial.print("\t");
    Serial.print(ay_offset);
    Serial.print("\t");
    Serial.print(az_offset);
    Serial.print("\t");
    Serial.print(gx_offset);
    Serial.print("\t");
    Serial.print(gy_offset);
    Serial.print("\t");
    Serial.println(gz_offset);
```

```
    Serial.println("\nData is printed as: acelX acelY acelZ giroX
giroY giroZ");
    Serial.println("Check that your sensor readings are close to 0 0
16384 0 0 0");
    Serial.println("If calibration was succesful write down your
offsets so you can set them in your projects using something similar
to mpu.setXAccelOffset(youroffset)");
    while (1);
  }
}

/////////////////////////////////////   FUNCTIONS
///////////////////////////////////////
void meansensors(){
  long
i=0,buff_ax=0,buff_ay=0,buff_az=0,buff_gx=0,buff_gy=0,buff_gz=0;

  while (i<(buffersize+101)){
    // read raw accel/gyro measurements from device
    accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

    if (i>100 && i<=(buffersize+100)){ //First 100 measures are
discarded
      buff_ax=buff_ax+ax;
      buff_ay=buff_ay+ay;
      buff_az=buff_az+az;
      buff_gx=buff_gx+gx;
      buff_gy=buff_gy+gy;
      buff_gz=buff_gz+gz;
    }
    if (i==(buffersize+100)){
      mean_ax=buff_ax/buffersize;
      mean_ay=buff_ay/buffersize;
      mean_az=buff_az/buffersize;
      mean_gx=buff_gx/buffersize;
      mean_gy=buff_gy/buffersize;
      mean_gz=buff_gz/buffersize;
    }
    i++;
    delay(2); //Needed so we don't get repeated measures
  }
}

void calibration(){
  ax_offset=-mean_ax/8;
  ay_offset=-mean_ay/8;
  az_offset=(16384-mean_az)/8;

  gx_offset=-mean_gx/4;
  gy_offset=-mean_gy/4;
  gz_offset=-mean_gz/4;
  while (1){
    int ready=0;
    accelgyro.setXAccelOffset(ax_offset);
    accelgyro.setYAccelOffset(ay_offset);
    accelgyro.setZAccelOffset(az_offset);

    accelgyro.setXGyroOffset(gx_offset);
    accelgyro.setYGyroOffset(gy_offset);
    accelgyro.setZGyroOffset(gz_offset);
```

```
    meansensors();
    Serial.println("...");

    if (abs(mean_ax)<=acel_deadzone) ready++;
    else ax_offset=ax_offset-mean_ax/acel_deadzone;

    if (abs(mean_ay)<=acel_deadzone) ready++;
    else ay_offset=ay_offset-mean_ay/acel_deadzone;

    if (abs(16384-mean_az)<=acel_deadzone) ready++;
    else az_offset=az_offset+(16384-mean_az)/acel_deadzone;

    if (abs(mean_gx)<=giro_deadzone) ready++;
    else gx_offset=gx_offset-mean_gx/(giro_deadzone+1);

    if (abs(mean_gy)<=giro_deadzone) ready++;
    else gy_offset=gy_offset-mean_gy/(giro_deadzone+1);

    if (abs(mean_gz)<=giro_deadzone) ready++;
    else gz_offset=gz_offset-mean_gz/(giro_deadzone+1);

    if (ready==6) break;
  }
}
```

Παράρτημα Γ

```
#include <EEPROM.h>

const int16_t savedLetters[][14] = {
    {637, 452, 553, 489, 511, 602, 608, 539, 0, 0, 0, 0, 86, -3},
    {481, 581, 642, 675, 611, 670, 659, 598, 0, 0, 0, 0, 77, -2},
    {626, 578, 580, 508, 485, 624, 552, 549, 0, 0, 0, 0, -73, 5},
    {589, 565, 627, 563, 514, 624, 581, 535, 0, 8, 0, 0, 81, -2},
    {463, 457, 621, 474, 504, 626, 603, 510, 0, 0, 0, 0, 75, -3},
    {481, 572, 624, 589, 504, 636, 583, 606, 0, 0, 0, 0, 10, 79},
    {475, 573, 609, 462, 504, 600, 596, 602, 0, 0, 0, 0, 82, -7},
    {488, 588, 634, 582, 606, 555, 635, 548, 0, 0, 0, 0, 15, 71},
    {453, 395, 526, 517, 517, 614, 576, 604, 0, 0, 0, 0, 77, -12},
    {626, 570, 627, 591, 494, 622, 582, 540, 0, 0, 0, 0, 78, -4},
    {555, 593, 623, 572, 600, 629, 601, 572, 0, 0, 0, 0, -64, 13},
    {516, 582, 623, 571, 597, 672, 659, 568, 0, 0, 0, 0, -72, 14},
    {539, 590, 594, 574, 587, 646, 590, 559, 0, 0, 0, 0, -72, 5},
    {491, 575, 646, 577, 609, 671, 662, 554, 0, 0, 0, 0, 10, 70},
    {507, 465, 537, 516, 512, 611, 611, 527, 0, 0, 0, 0, 64, -5},
    {528, 580, 629, 501, 485, 626, 593, 612, 0, 0, 0, 0, -62, 22},
    {554, 597, 564, 567, 616, 586, 613, 558, 0, 8, 0, 0, 77, -11},
    {587, 467, 544, 494, 590, 610, 582, 521, 8, 8, 0, 0, 86, -3},
    {522, 585, 633, 438, 517, 588, 620, 593, 0, 0, 0, 0, -61, 20},
    {620, 462, 550, 583, 492, 626, 568, 604, 0, 0, 0, 0, 80, -8},
    {641, 520, 527, 586, 601, 668, 662, 606, 8, 0, 0, 0, 82, -3},
    {451, 528, 614, 523, 604, 567, 602, 550, 0, 0, 0, 0, 82, -3},
    {530, 579, 584, 584, 561, 665, 644, 610, 0, 0, 0, 8, 83, -3},
    {632, 547, 554, 589, 527, 653, 613, 598, 0, 8, 8, 0, 49, 8}

};
void setup() {
    // put your setup code here, to run once:
    Serial.begin(115200);
    //EEPROM.write(256,11);

    int l = 0, k = 0;
    int eeAddress = 0;
    for (int i = 0; i < 336; i++) {


        EEPROM_writeAnything(eeAddress,savedLetters[k][l]);
        Serial.print(String(k) + " "+ String(l) + " " );
        Serial.print(String(savedLetters[k][l]) + " ");

        // Serial.println(String((int8_t)EEPROM.read(eeAddress) *
8));
        //int16_t val;
        //EEPROM_readAnything(eeAddress,val);
        //val = val + 10;
//Serial.println(String(val));

        eeAddress = eeAddress + 2;

        l++;
        if (l == 14) {
            k++;
            l = 0;
        }
```

```
    }
}

void loop() {
    // put your main code here, to run repeatedly:

}

template <class T> int EEPROM_writeAnything(int ee, const T& value)
{
    const byte* p = (const byte*)(const void*)&value;
    int i(0);
    for (; i < sizeof(value); ++i)
        EEPROM.write(ee++, *p++);
    return i;
}


template <class T> int EEPROM_readAnything(int ee, T& value)
{
    byte* p = (byte*)(void*)&value;
    int i (0);
    for (; i < sizeof(value); ++i)
        *p++ = EEPROM.read(ee++);
    return i;
}
```

Παράρτημα Δ Αναλυτικό κόστος

Αρ.	Υλικά	ΠΟΣΟΤΗΤΑ	ΤΙΜΗ	ΤΙΜΗ + ΦΠΑ	ΤΙΜΗ * ΠΟΣΟΤΗΤΑ	ΜΕΤΑΦΟΡΙΚΑ	ΣΥΝΟΛΟ
1	Arduino UNO *	1	7,05	7,05	7,05	0	7,05
2	flex sensor 2"	8	7,5	9,3	74,4	0	74,4
3	Accelerometer/ Gyroscope	1	1,51	1,51	1,51	0	1,51
4	Contact sensors π.χ FSR-400	5	6,53	8,1	40,5	0	40,5
5	Communicaiton Module	1	3,27	3,27	3,27	0	3,27
6	MCP3008 ADC	1	3,84	3,84	3,84	2,95	6,79
7	Battery Holder	1	0,76	0,76	0,76	0	0,76
9	breadboard & Jumper wires	1	3,66	3,66	3,66	0	3,66
10	Resistors Bulk	1	4,61	4,61	4,61	0	4,61
						ΣΥΝΟΛΙΚΑ	**142.55**

Πίνακας 7. Συνολικό κόστος συσκευής