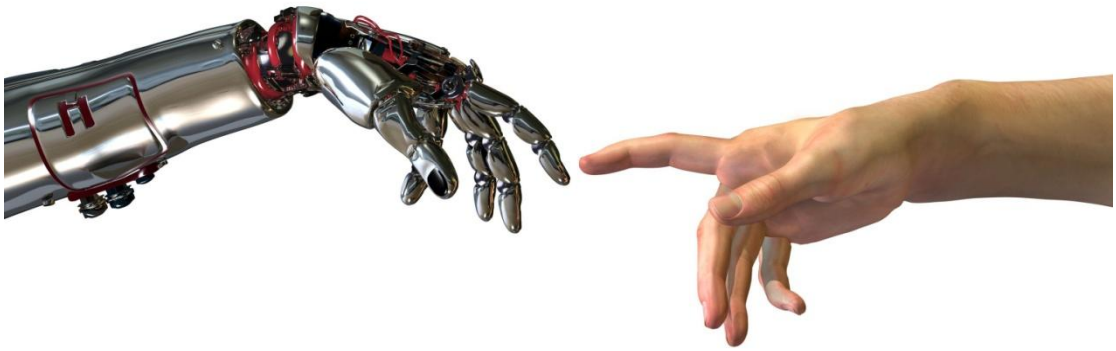
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ, ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΣΤΗ ΡΟΜΠΟΤΙΚΗ



Διπλωματική εργασία

ΑΝΑΠΤΥΞΗ ΣΥΣΤΗΜΑΤΟΣ ΑΠΤΙΚΟΥ ΓΑΝΤΙΟΥ – ΡΟΜΠΟΤΙΚΟΥ ΧΕΡΙΟΥ ΓΙΑ ΜΙΜΗΣΗ ΤΩΝ ΚΙΝΗΣΕΩΝ ΤΟΥ ΑΝΘΡΩΠΙΝΟΥ ΧΕΡΙΟΥ ΜΕΣΩ ΑΠΤΙΚΩΝ ΚΑΙ ΟΠΤΙΚΩΝ ΑΙΣΘΗΤΗΡΩΝ

του ΛΑΤΣΟΥ ΗΡΑΚΛΗ (Α.Μ. 4)

Επιβλέπων καθηγητής: κος Καζαρλής Σπυρίδων

Υποβλήθηκε ως απαιτούμενο για την απόκτηση του Μεταπτυχιακού Διπλώματος στη Ρομποτική

Σέρρες, Δεκέμβριος 2020

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ

Βεβαιώνω ότι είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην διπλωματική εργασία. Επίσης, έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης, βεβαιώνω ότι αυτή η διπλωματική εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Μεταπτυχιακού με τίτλο «Ρομποτική» (MSc in Robotics) του Τμήματος Μηχανικών Πληροφορικής Τ.Ε. της Σχολής Τεχνολογικών Εφαρμογών του Τ.Ε.Ι. Κεντρικής Μακεδονίας.

Ο Δηλών,

Λάτσος Ηρακλής

ΕΥΧΑΡΙΣΤΙΕΣ

Με την ολοκλήρωση της διπλωματικής μου εργασίας, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Σπυρίδωνα Καζαρλή για την υποστήριξη και εμπιστοσύνη που μου έδειξε εξ΄ αρχής με την ανάθεση του προτεινόμενου θέματος.

Τέλος, θα ήθελα να ευχαριστήσω τη σύζυγό μου για την κατανόηση, υπομονή και ηθική στήριξη της, για την εκπλήρωση της διπλωματικής μου εργασίας.

Λάτσος Ηρακλής

Περιεχόμενα

Πίνακας περιεχομένων εικόνων

Κατάλογος των πινάκων

Κατάλογος των διαγραμμάτων

Περίληψη

Στην παρούσα διπλωματική εργασία, μελετήσαμε τον έλεγχο ενός ρομποτικού χεριού μέσω ενός απτικού γαντιού με αισθητήρες κάμψης. Μελετήθηκε επίσης, ο ασύρματος τρόπος μετάδοσης των σημάτων των αισθητήρων κάμψης, που τοποθετήθηκαν σε κάθε ένα από τα 5 δάχτυλα του γαντιού μέσω Bluetooth επικοινωνίας. Παράλληλα, θα πρέπει να ενημερώνεται ο χρήστης με οπτική ένδειξη εάν έχει διακοπεί ή όχι η επικοινωνία μεταξύ του γαντιού και της βάσης πάνω στην οποία βρίσκεται το ρομποτικό χέρι.

Επιπροσθέτως, μελετήσαμε την μίμηση των κινήσεων ενός ανθρώπινου χεριού μέσω οπτικού μέσου (κάμερας). Με τη χρήση ενός Raspberry Pi που εκτελεί ένα python script και με τη βοήθεια της βιβλιοθήκης OpenCV, ανιχνεύουμε το χέρι, πόσα δάχτυλα εμφανίζονται εντός της κάμερας, καθώς και ποια είναι αυτά μέσα από 13 στο σύνολο περιπτώσεις. Στη συνέχεια, στέλνονται οι κατάλληλες εντολές σε έναν μικροελεγκτή Arduino ο οποίος κινεί τα δάχτυλα του ρομποτικού χεριού (μέσω του PCA9685 servo driver) με τη χρήση σειριακού πρωτοκόλλου επικοινωνίας.

Η επιλογή μεταξύ του πρώτου και του δεύτερου τρόπου λειτουργίας γίνεται μέσω ενός επιλογικού διακόπτη ¨0-1-2¨ όπου στη θέση 0 το ρομποτικό χέρι είναι κλειστό, στην θέση 1 εκτελεί τον έλεγχο μέσω του γαντιού, ενώ στη θέση 2 εκτελεί τον έλεγχο μέσω κάμερας.

Στα πλαίσια μελέτης των 2 ανωτέρω τρόπων ελέγχου του ρομποτικού χεριού, κατασκευάσαμε μια φορητή βάση όπου εντός της περιέχει το ρομποτικό χέρι, οθόνη LCD για απεικόνιση, κάμερα, μικροελεγκτής Arduino, Raspberry Pi, PCA9685 servo driver, Bluetooth module, τροφοδοτικά καθώς και άλλα επιμέρους υλικά.

Abstract

In the current thesis, we studied the control of a robotic hand through the usage of a custom built haptic glove with flex sensors. We also studied the wireless way of transmitting the flex sensor signals that were put in each of the 5 fingers of the custom built glove. In addition, the user should be informed with an optical indication, if the wireless communication between the glove and the base on top of which, the robotic hand is placed, is interrupted or not.

Moreover, we studied the imitation of the movements of a human hand with the use of a camera. A Raspberry Pi that ‘’runs’’ a python script together with the help of the OpenCV library, detects the hand, counts how many fingers are displayed in front of the camera and calculates which one of the fingers are, from a total of 13 cases. Then, the suitable commands are sent to an Arduino microcontroller who moves the robotic hands fingers (through a PCA9685 servo driver) with the use of serial communication protocol.

The choice between the first and second mode of operation is done by a ¨0-1-2¨ rotary switch, that in the 0 position the robotic hand is closed, in position 1 the glove control mode is chosen and in position 2 the chosen mode is the camera control.

Within the study of the 2 control modes of the robotic hand mentioned, we built a base inside in which are placed the robotic hand, an LCD screen for display purposes, a camera, an Arduino microcontroller, a Raspberry Pi, a PCA9685 servo driver, a Bluetooth module, power supplies and other individual materials.

Συνεισφορά της εργασίας

Η συνεισφορά της συγκεκριμένης διπλωματικής εργασίας εντοπίζεται στα παρακάτω

σημεία. Πρώτον στην αναγνώριση και μίμηση των κινήσεων του ανθρώπινου χεριού, αναπτύχθηκε αλγόριθμος ο οποίος κάνει σχετικές συγκρίσεις βασισμένος και στη γεωμετρία που σχηματίζουν τα δάχτυλα του ανθρώπινου χεριού. Επίσης, εάν η θερμοκρασία του επεξεργαστή ξεπεράσει τους 85°C τερματίζεται η εκτέλεση του προγράμματος και εμφανίζουμε στην οθόνη (terminal) σχετικό μήνυμα. Η ακρίβεια ήταν αρκετά ικανοποιητική εφ' όσον τηρηθούν οι σχετικές προδιαγραφές όπως είναι η δημιουργία ενός πλαισίου όπου ο χρήστης θα τοποθετεί το δεξί του χέρι ενώ παράλληλα θα πρέπει να υπάρχουν σαφώς και τα κατάλληλα επίπεδα φωτεινότητας στο χώρο. Στο κεφάλαιο **4.3.3** – **4.3.4** εξηγείται η δομή του αλγορίθμου.

Δεύτερον, πραγματοποιήθηκαν και οι κάτωθι υλοποιήσεις:

- Υλοποίηση του απτικού γαντιού με σύνδεση των αντιστάσεων κάμψης σε μικροελεγκτή arduino Nano με διαιρέτες τάσεις καθώς επίσης και σύνδεση ενός module ασύρματης επικοινωνίας Bluetooth.

- Υλοποίηση Kalman φίλτρου για το φιλτράρισμα των αναλογικών τιμών των αισθητήρων κάμψης προτού αποσταλθούν μέσω Bluetooth.

- Υλοποίηση αλγορίθμου όπου εάν χαθεί η ασύρματη Bluetooth επικοινωνία μεταξύ ενός master και ενός slave, να μην έχουμε αποσυγχρονισμό και λαμβάνονται τα πακέτα σε λάθος σειρά. Αυτό μπορεί να συμβεί κυρίως στην περίπτωση όπου τα Bluetooth modules (τα οποία έχουν εμβέλεια περίπου 10 μέτρα) είναι εκτός εμβέλειας. Η λειτουργία αυτή εξηγείται στο κεφάλαιο **4.2.10**.

- Υλοποίηση ελέγχου των σερβοκινητήρων του ρομποτικού χεριού μέσω του PCA9685 servo driver και της κατάλληλης βιβλιοθήκης.

- Υλοποίηση σειριακής επικοινωνίας μεταξύ Raspberry και Arduino για την κωδικοποίηση των κινήσεων των δαχτύλων

- Υλοποίηση μεταγωγικού διακόπτη για σύνδεση/αποσύνδεση των σημάτων Receive/ Transmit (Rx/Tx) μεταξύ του Arduino, του Raspberry Pi και του HC 05 bluetooth

module που λαμβάνει τα ασύρματα δεδομένα του γαντιού. Ο μεταγωγικός αυτός διακόπτης υλοποιήθηκε με τη χρήση NPN τρανζίστορ, διόδου και ενός DC ρελέ με κανονικά ανοιχτές και κλειστές επαφές (N.O / N.C)

- Υλοποίηση φορητής κατασκευής για το ρομποτικό χέρι και σύνδεση των επιμέρους εξαρτημάτων ελέγχου όπως ελεγκτές, κύκλωμα ισχύος, κάμερα, LCD οθόνη απεικόνισης κ.α.

Κεφάλαιο 1- Εισαγωγή

1.1 Εισαγωγικές παρατηρήσεις

1.1.1 Απτικά γάντια

Η χρήση απτικών γαντιών δεν είναι καινούρια ιδέα καθώς ήδη κατά τη δεκαετία του 80’ χρησιμοποιούνται στον τομέα των βιντεοπαιχνιδιών και ακόμη παλαιότερα στον στρατιωτικό τομέα της αεροπορίας. Τα τελευταία χρόνια όμως, η χρήση τους έχει εξαπλωθεί και σε άλλους τομείς όπως η εικονική πραγματικότητα, ο έλεγχος απομακρυσμένων συσκευών και ρομποτικών συσκευών, σε συστήματα βοήθειας τυφλών και κωφών ατόμων καθώς και στην ιατρική και οδοντιατρική [Δ1].

Η λέξη απτικός αναφέρεται στην αίσθηση της αφής του χρήστη για τον έλεγχο και την αλληλεπίδραση με άλλους υπολογιστές. Η απτική τεχνολογία χρησιμοποιείται κυρίως στη δημιουργία και έλεγχο εικονικών αντικειμένων καθώς και στον απομακρυσμένο έλεγχο συσκευών. Μερικά είδη hardware για την παροχή απτικής ανάδρασης περιλαμβάνουν τη χρήση δονητικών κινητήρων (Eccentric Rotation Mass), πιεζοηλεκτρικούς ενεργοποιητές (piezoelectric actuators) και άλλες τεχνολογίες.

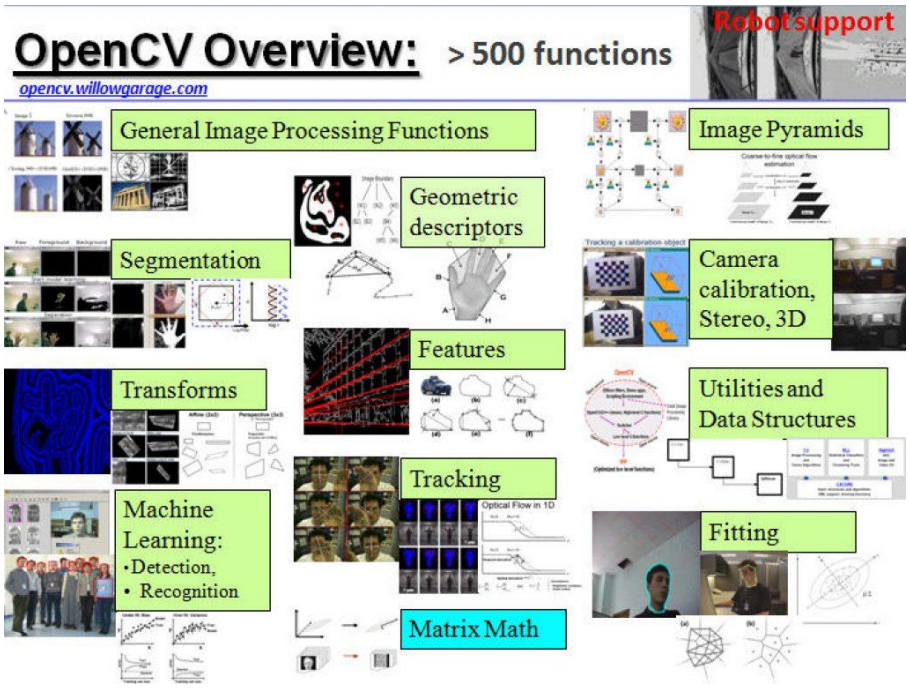
Εικόνα 1: Απτικός εξοπλισμός για εικονική πραγματικότητα

(Πηγή: https://www.vrfocus.com/2019/06/manus-vr-launches-e5000-haptic-gloves-for-enterprise/)

Ένας ακόμη σημαντικός τομέας χρήσης των απτικών γαντιών είναι στην εκπαίδευση εργαζομένων σε κλάδους όπως η πυρόσβεση, η γραμμή παραγωγής σε ένα εργοστάσιο, η χειρουργική, η συντήρηση και ασφάλεια σε πυρηνικό εργοστάσιο καθώς και άλλους τομείς [Δ2].

1.1.2 Μηχανική όραση

Η χρήση καμερών για μηχανική όραση αυξάνεται διαρκώς με ταχύτατους ρυθμούς και κατέχουν θέση σε πληθώρα εφαρμογών όπως στην αυτοκίνηση, σε συστήματα ασφαλείας, ρομποτικές εφαρμογές (αυτόνομες και μη), στη βιομηχανία για ποιοτικό έλεγχο, στη γεωργία, κτηνοτροφία καθώς και σε πολλές άλλες ακόμα [Δ4].



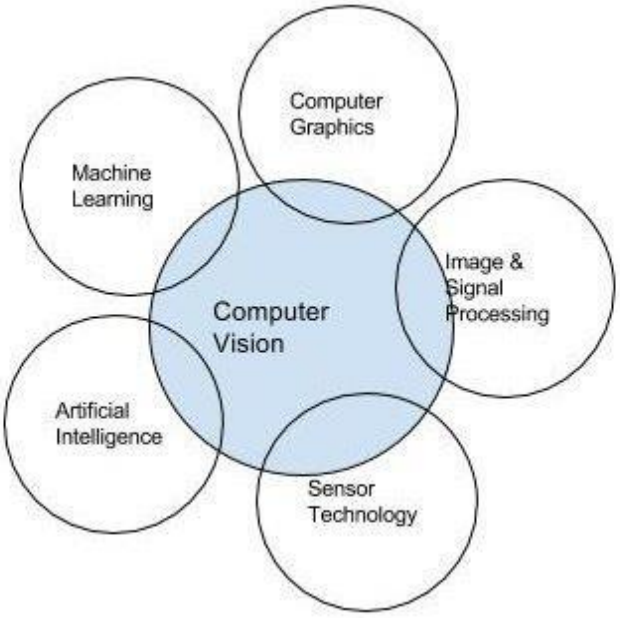
Εικόνα 2: Λειτουργίες της OpenCV

(Πηγή: https://www.edge-ai-vision.com/2011/11/introduction-to-computer-vision-using-opencv-article/)

Επειδή τα συστήματα μηχανικής όρασης επεξεργάζονται εικόνες ακόμη και υψηλής ευκρίνειας, η χρήση τους ήταν περιορισμένη παλαιότερα λόγω των υψηλών απαιτήσεων σε πόρους καθώς και σε επεξεργαστική ισχύ, γεγονός που δεν υφίσταται πλέον λόγω της ραγδαίας ανάπτυξης του hardware των υπολογιστών και των ενσωματωμένων συστημάτων.

Παράλληλα, η ανάπτυξη της βιβλιοθήκης OpenCV επιτρέπει σε ολοένα και μεγαλύτερη μερίδα χρηστών (και εταιριών) την ανάπτυξη αλγορίθμων μηχανικής όρασης και εκμάθησης (machine vision, machine learning).

Ο όρος ‘’μηχανική όραση’’ ορίζει τις τεχνολογίες και τις μεθόδους που χρησιμοποιούνται για την εξαγωγή ενός συνόλου πληροφοριών από μια εικόνα. Σε αντίθεση, η επεξεργασία εικόνας δίνει ως έξοδο μια νέα εικόνα. Οι εξαγόμενες πληροφορίες μπορεί να είναι πολύπλοκες όπως π.χ η θέση, ο προσανατολισμός και άλλες ιδιότητες ενός αντικειμένου. Έπειτα οι πληροφορίες αυτές μπορούν να χρησιμοποιηθούνε για την αυτόματη ανίχνευση ελαττωματικών προϊόντων, για την καθοδήγηση ρομπότ καθώς και αυτόνομων οχημάτων. Η μηχανική όραση αποτελεί ακρογωνιαίο λίθο για την επίλυση πολλών προβλημάτων και συνδυάζεται με διάφορες τεχνολογικές μεθόδους όπως τα γραφικά Η/Υ, η μηχανική μάθηση, η τεχνητή νοημοσύνη (A.I) και άλλες, οι οποίες μπορούνε να συνοψιστούνε όπως στην παρακάτω εικόνα.



Εικόνα 3: Η μηχανική όραση και τομείς αλληλεπίδρασης της

(Πηγή: https://www.iotforall.com/computer-vision-iot)

1.2 Σκοπός εργασίας

Σκοπός της παρούσας εργασίας είναι αρχικά να γίνει μια μελέτη πάνω στον έλεγχο ενός ρομποτικού χεριού μέσα από την δημιουργία ενός απτικού γαντιού με αισθητήρες κάμψης. Μελετήθηκε, ο ασύρματος τρόπος μετάδοσης των σημάτων των αισθητήρων κάμψης, που τοποθετήθηκαν σε κάθε ένα από τα 5 δάχτυλα του γαντιού μέσω Bluetooth επικοινωνίας.

Επιπλέον, μελετήθηκε η μίμηση των κινήσεων ενός ανθρώπινου χεριού μέσω οπτικού μέσου (συγκεκριμένα κάμερας). Η μίμηση αυτή πραγματοποιήθηκε με τη χρήση ενός Raspberry Pi που εκτελεί ένα python script (με τη βοήθεια της βιβλιοθήκης OpenCV) ώστε, τα δάχτυλα που εμφανίζονται εντός της κάμερας, να αναγνωρίζονται από το λογισμικό καθώς και ποια είναι αυτά μέσα από 13 στο σύνολο περιπτώσεις.

1.3 Δομή εργασίας

Η παρούσα εργασία αποτελείται από πέντε συνολικά κεφάλαια. Πιο συγκεκριμένα, στο πρώτο κεφάλαιο πραγματοποιείται μια ανάλυση του σκοπού της εργασίας και έπειτα, παρουσιάζονται παραδείγματα της χρήσης παρόμοιων ρομποτικών συστημάτων στον τομέα της ιατρικής αλλά και σε άλλους τομείς.

Στο δεύτερο κεφάλαιο, γίνεται μια έρευνα πάνω στην θέση που διατηρούν άλλοι συγγραφείς σχετικά με το θέμα της απτικής τεχνολογίας αλλά και της μηχανικής όρασης.

Στο τρίτο κεφάλαιο, δίνονται κάποιοι ορισμοί που εμπερικλείουν τις έννοιες «Ρομποτικό χέρι», «Kalman φίλτρο», «Σερβοκινητήρες (Rc servos)», «HC05 Bluetooth module», «Pi camera v2», «Αισθητήρες κάμψης (Flex sensors)» αλλά και άλλες έννοιες που μας απασχόλησαν στην παρούσα διπλωματική εργασία.

Στο τέταρτο κεφάλαιο, παρουσιάζεται το πρακτικό κομμάτι της εργασίας. Πιο συγκεκριμένα, το παρόν κεφάλαιο αποτελεί μια παρουσίαση της όλης κατασκευαστικής και προγραμματιστικής εμπειρίας που αποκομίσθηκε στην διάρκεια της παρούσας εργασίας.

Στο πέμπτο και τελευταίο κεφάλαιο παρουσιάζονται τα συμπεράσματα που αποκομίστηκαν από την παρούσα εργασία, τα προβλήματα που συναντήθηκαν κατά την διάρκεια της υλοποίησης της, καθώς, και κάποιοι προβληματισμοί για μελλοντική έρευνα.

1.4 Περιγραφή του προβλήματος

Κατά τη διάρκεια της διπλωματικής εργασίας μελετήσαμε και κατασκευάσαμε το απτικό γάντι καθώς και τη βάση με την κάμερα για τον έλεγχο του ρομποτικού χεριού.

Το πρόβλημα ουσιαστικά αποτελείται από 2 ανεξάρτητα μέρη (modes ελέγχου) όπου στο πρώτο το απτικό γάντι ελέγχει ασύρματα το ρομποτικό χέρι, ενώ στο δεύτερο μια κάμερα που έχει τοποθετηθεί στην βάση, κάνει αναγνώριση του δεξιού ανθρώπινου χεριού (χωρίς γάντι) και αντίστοιχα κινεί το ρομποτικό χέρι. Ο αλγόριθμος που αναπτύχθηκε αναγνωρίζει έως 13 περιπτώσεις χειρονομιών (gestures). Η επιλογή του κάθε mode ελέγχου γίνεται μέσω ενός επιλογικού διακόπτη.

Στο απτικό γάντι έχουν τοποθετηθεί αισθητήρες κάμψης που αλλάζουν τιμή ανάλογα με την κλίση του κάθε δαχτύλου. Αφού φτιάξουμε το ηλεκτρολογικό κύκλωμα για την τροφοδοσία του μικροελεγκτή και των επιμέρους κυκλωμάτων πρέπει να επιλέξουμε τον ασύρματο τρόπο μετάδοσης των πακέτων με τις τιμές των δαχτύλων στη βάση που θα ελέγχει το ρομποτικό χέρι. Επιπλέον, επειδή οι αναλογικές τιμές των δαχτύλων είναι ασταθείς, πρέπει να επιλέξουμε τον κατάλληλο τρόπο να φιλτράρουμε και έπειτα να αποστείλουμε ασύρματα αυτές τις τιμές.

Η κατασκευή της βάσης έθεσε τον προβληματισμό του μεγέθους καθώς έπρεπε να είναι μικρή σε διαστάσεις για λόγους ευκολίας στη μετακίνηση και στη χρήση. Επίσης η τροφοδοσία των επιμέρους κυκλωμάτων καθώς και του μικροελεγκτή και του Raspberry έπρεπε να επιλεχθεί σύμφωνα με το μικρότερο δυνατό σε μέγεθος τροφοδοτικό το οποίο όμως θα έπρεπε παράλληλα να παρέχει και το απαιτούμενο ρεύμα.

Όσον αφορά την ασύρματη επικοινωνία του μικροελεγκτή της βάσης, έπρεπε να αναπτυχθεί αλγόριθμος ο οποίος να γνωρίζει πότε μεταδίδεται κάθε νέο πακέτο με τις 5 τιμές των δαχτύλων και να στέλνει πίσω στον μικροελεγκτή του απτικού γαντιού μια επιβεβαίωση ότι έχει λάβει το πακέτο. Ο χρήστης του απτικού γαντιού θα πρέπει να γνωρίζει ανά πάσα στιγμή εάν υπάρχει ασύρματη επικοινωνία του γαντιού με τη βάση ή εάν έχει διακοπεί.

Ο ‘’εγκέφαλος’’ της διεργασίας (Raspberry) πρέπει να είναι ικανός να εκτελεί και να στέλνει σε έναν μικροελεγκτή Arduino (low level control) τις κατάλληλες εντολές κίνησης και να μην επιβαρύνεται περισσότερο. Ο τρόπος επικοινωνίας του Raspberry με τον Arduino της βάσης

είναι ένα ακόμα πρόβλημα που αντιμετωπίσαμε για την υλοποίηση της συγκεκριμένης διπλωματικής εργασίας.

Ο κάθε χρήστης πρέπει να έχει μια διεπαφή ή έστω ενημέρωση όσο εκτελείται ο αλγόριθμος για την ανίχνευση μέσω κάμερας, των επιπέδων φωτεινότητας, του αριθμού των δαχτύλων που ανιχνεύτηκαν, της ονομασίας αυτών καθώς και οποιαδήποτε άλλης πληροφορίας βοηθάει το χρήστη.

Το πρώτο ερώτημα το οποίο πρέπει να απαντήσουμε στον έλεγχο του ρομποτικού χεριού μέσω κάμερας είναι το εξής: ‘’ Ποιοι είναι οι τρόποι ανίχνευσης ενός αντικειμένου που ενδιαφερόμαστε σε μια στατική κατασκευή;’’

Ως γνωστόν όλες οι κάμερες είναι εξαρτημένες από τα επίπεδα και τον τύπο φωτεινότητας του χώρου (υπάρχει διαφορά εάν η κάμερα φωτίζεται από φυσικό ή τεχνητό φως από κάποιον προβολέα) καθώς και από τις σκιάσεις, με συνέπεια η εύρεση των κατάλληλων συνθηκών φωτεινότητας να απαιτεί ευρύ πεδίο δοκιμών και πειραματισμών.

Η αναγνώριση του αριθμού των δαχτύλων που βλέπει η κάμερα αποτελεί ένα μέρος του προβλήματος για τον έλεγχο του ρομποτικού χεριού μέσω κάμερας, ενώ η αναγνώριση και η εύρεση ποιων δαχτύλων συγκεκριμένα ήταν ένα δεύτερο μέρος του προβλήματος ιδιαίτερα απαιτητικού στη λύση.

Τα οφέλη και αποτελέσματα που αναμένουμε με την εκπλήρωση της διπλωματικής εργασίας συνοψίζονται στα εξής σημεία:

- ✓ Στην απόκτηση τεχνογνωσίας για την ασύρματη αμφίδρομη επικοινωνία μεταξύ ενός master και ενός slave μικροελεγκτή.
- ✓ Στην καταγραφή της βελτίωσης των κινήσεων με την εφαρμογή ενός φίλτρου όπως είναι το Kalman το οποίο χρησιμοποιείται ευρέως εδώ και αρκετές δεκαετίες σε πληθώρα εφαρμογών.
- ✓ Στην κατανόηση της διασύνδεσης ενός σημαντικού αριθμού ανεξάρτητων εξαρτημάτων μεταξύ τους καθώς και οι περιορισμοί τους.
- ✓ Στην τεχνογνωσία που αποκτήθηκε στη χρήση της OpenCV και των λειτουργιών της.
- ✓ Στην κατανόηση της επίδρασης της φωτεινότητας σε συστήματα με κάμερες.

- ✓ Στον τρόπο προσέγγισης προβλημάτων ανίχνευσης ενός στοιχείου ενδιαφέροντος όπως π.χ ένα ανθρώπινο χέρι ή ένα οποιοδήποτε αντικείμενο.

Κεφάλαιο 2- Επισκόπηση βιβλιογραφίας

Το παρόν κεφάλαιο αφορά στη μελέτη που πραγματοποιήθηκε για τον προσδιορισμό παρόμοιων θεμάτων απτικού γαντιού, και μπορούμε να κατανοήσουμε τον τρόπο που προσεγγίζουν το υπό μελέτη θέμα διαφορετικοί συγγραφείς μέσα από βιβλία και επιστημονικές εφημερίδες.

2.1 Εισαγωγή στην απτική τεχνολογία

Η απτική ορίζεται ως η αίσθηση της αφής και αφορά την εξομοίωση των ασκούμενων δυνάμεων, των δονήσεων και των κινήσεων διαφόρων εικονικών αντικειμένων *(Augmented Reality" (PDF). Zums.ac.ir. Retrieved 19 April 2019 [B7]).* Οι απτικές συσκευές διαθέτουνε απτικούς αισθητήρες (tactile sensors) που μετράνε δυνάμεις ασκούμενες από το χρήστη. Οι πιο απλές απτικές συσκευές είναι τα χειριστήρια και τιμόνια παιχνιδομηχανών.

Η απτική τεχνολογία μελετάει τον τρόπο με τον οποίο η ανθρώπινη αίσθηση αφής μπορεί να συνδεθεί με τον έλεγχο εικονικών αντικειμένων. Οι περισσότεροι ερευνητές διακρίνουν 3 τύπους αισθητήριων συστημάτων που σχετίζονται με την αφή: α) δερματικά β) κιναισθητικά και γ) απτικά [Δ3]. Η αίσθηση της αφής μπορεί να διαχωριστεί σε 2 μέρη, ως παθητική και ενεργητική και ο όρος ‘’απτική’’ συμπεριλαμβάνεται στην ενεργητική αφή για την επικοινωνία και αναγνώριση αντικειμένων.

2.1.1 Υλοποίηση μιας απτικής συσκευής

Η υλοποίηση μιας συσκευής με απτική τεχνολογία μπορεί να επιτευχθεί με τους παρακάτω τρόπους.

A) <u>Δόνηση</u>

Αποτελεί τον πιο διαδεδομένο τρόπο υλοποίησης μιας απτικής συσκευής και συνήθως περιλαμβάνει έναν κινητήρα με προσαρμοσμένη έκκεντρη περιστρεφόμενη μάζα (Eccentric Rotating Mass). Καθώς ο ρότορας του κινητήρα περιστρέφεται, περιστρέφεται και η αναφερόμενη μάζα και έχει συνέπεια τη δημιουργία δονήσεων. Τελευταία, νεότερες συσκευές δημιουργούνε δονήσεις μέσα από ευθύγραμμους ενεργοποιητές συντονισμού (Linear Resonant Actuators) όπου η μάζα κινείται παλινδρομικά και όχι περιστροφικά, πετυχαίνοντας ταχύτερες αποκρίσεις. Επιπλέον,

υπάρχουν και οι πιεζοηλεκτρικοί ενεργοποιητές οι οποίοι έχουν ακόμη πιο αποτελεσματική απόκριση από τους LRA ενεργοποιητές, αλλά απαιτούνε μεγαλύτερες τάσεις λειτουργίας από τους δυο προηγούμενους ενεργοποιητές [Δ5].

Eccentric rotary mass (ERM)	Linear resonant actuators (LRA)	Piezo disks and benders
Off-center rotating mass spins, creating an omni-directional vibration	Magnet attached to a spring, surrounded by a coil and housed in a casing, moves in a linear fashion and brought up to the resonant frequency	Piezoelectric ceramic material that expands or contracts when voltage is applied
Modulation limited to pulsing and speed	Modulation limited to amplitude of input at resonant frequency	Frequency and amplitude can be modulated freely
Vibration of entire device (global)	Vibration of entire device (global)	Vibration of entire device (global) and more localized haptics

Εικόνα 4: Κατηγορίες ενεργοποιητών

(Πηγή:https://www.electronicdesign.com/technologies/components/article/21802172/piezo-disks-deliver-vertical-haptic-feedback)

Β) Ανάδραση δύναμης

Μερικές συσκευές χρησιμοποιούνε σερβοκινητήρες για να εξομοιώσουνε την αλληλεπίδραση των επιμέρους δυνάμεων. Συναντώνται αρκετά συχνά σε τιμόνια εξομοιωτών.

Γ) Υπερήχους

Με τη χρήση στοχευμένων υπερήχων μπορεί ένας χρήστης να αισθανθεί ‘’πίεση’’ χωρίς την άμεση επαφή του με ένα αντικείμενο. Η αίσθηση της ‘’πίεσης’’ δημιουργείται από ειδικούς μετατροπείς (transducers) η οποίοι θα μεταβάλλουν την φάση και ένταση των στοχευμένων υπερήχων.

Εικόνα 5: Απτική συσκευή υπερήχων

(Πηγή: https://www.extremetech.com/extreme/195394-3d-shapes-with-ultrasound)

Δ) Με δακτυλίους δίνης αέρα

Τα δαχτυλίδια δίνης αέρα είναι ουσιαστικά μικρές συγκεντρωμένες ριπές αέρα τις οποίες μπορούμε να χρησιμοποιήσουμε για να επιτευχθεί απτική ανάδραση χωρίς επαφή. Μερικές από τις εταιρίες που έχουν πειραματιστεί με αυτήν την μέθοδο είναι η Microsoft και η Disney Research.

2.1.3 Κατηγορίες

Ο άνθρωπος στηρίζεται διαρκώς στην απτική αίσθηση με τρόπους που πολλές φορές δεν γίνεται αντιληπτό. Τα νεύρα, οι αρθρώσεις, οι μύες και τα όργανα μας λένε τι θέση έχει το σώμα μας, πόσο σφιχτά κρατάμε ένα αντικείμενο κτλ. Ένα μεγάλο πλήθος μηχανικών από όλο τον πλανήτη εργάζεται στο κομμάτι της δημιουργίας ρεαλιστικών αισθήσεων αφής το οποίο θα επιτρέπει καλύτερο έλεγχο ρομποτικών συστημάτων, βελτίωση στους τομείς της εκπαίδευσης, της επικοινωνίας, της εικονικής πραγματικότητας, ακόμη και στη φυσική αποκατάσταση [B4].

Παλαιότερα, η απτική ήταν αποδοτική ώστε ο χρήστης να καταλαβαίνει όταν έκανε μια συγκεκριμένη κίνηση (π.χ μέσω δόνησης) αλλά τα τελευταία χρόνια η τεχνολογία της απτικής εστιάζει στο να υπάρχει μια φυσική αίσθηση και αλληλεπίδραση.

Οι απτικές συσκευές μπορούνε να χωριστούνε σε 3 κύριες κατηγορίες: μέσω άμεσης ή έμμεσης αφής (graspable), μέσω κάποιου γαντιού ή οτιδήποτε μπορούμε να φορέσουμε (wearable) και μέσω αγγίγματος σε κάποια ειδική επιφάνεια (touchable) ή οθόνη. Στην πρώτη

περίπτωση υπάρχουν τα χειριστήρια τα οποία προβάλλουν κατάλληλα επίπεδα αντίστασης στον χρήστη, στην δεύτερη περίπτωση ανήκουν τα απτικά γάντια ενώ στην τρίτη περίπτωση ανήκουν οι επιφάνειες όπως οθόνες οι οποίες δονούνται όταν ο χρήστης κάνει μια συγκεκριμένη ενέργεια.

Εικόνα 6: Κατηγορίες απτικών συσκευών

(Πηγή: https://www.smithsonianmag.com/innovation/heres-what-future-haptic-technology-looks-or-rather-feels-180971097/)

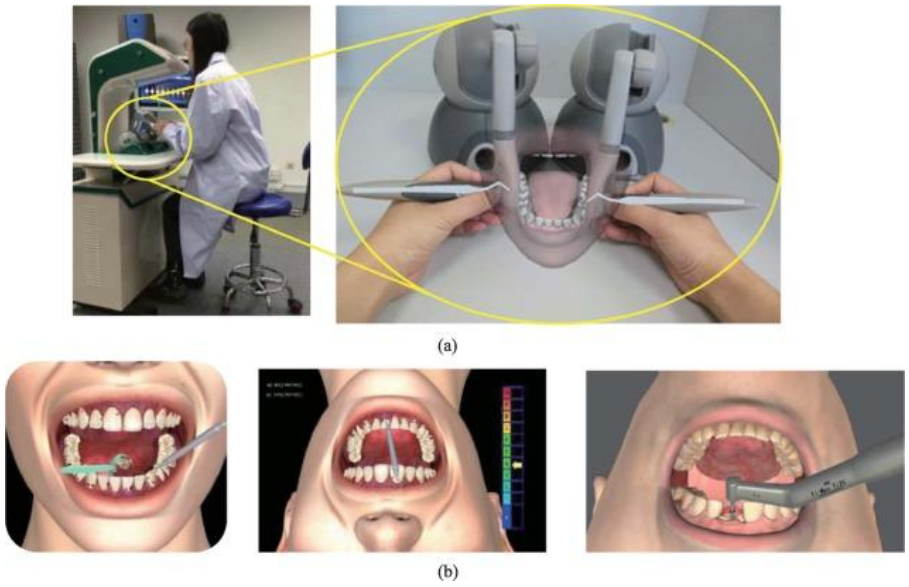
2.1.3.1 Τυπικές εφαρμογές

Μερικές τυπικές εφαρμογές απτικών συστημάτων είναι η εικονική χειρουργική, η μηχανολογική κατασκευή, εφαρμογές βασισμένες σε εργαλεία χειρός, στα βιντεοπαιχνίδια, στους Η/Υ, σε φορητές συσκευές, στη ρομποτική, στην εικονική πραγματικότητα, στις τέχνες στην αεροναυπηγική καθώς και σε άλλες εφαρμογές. Υπάρχουν βεβαίως πολλές προκλήσεις στα απτικά συστήματα αφής όπως για παράδειγμα πως αποδίδουμε την αίσθηση του βάρους ενός αντικειμένου όταν αγγίζουμε ένα ψηφιακό χωρίς βάρος αντικείμενο; Μελετώντας την νευροεπιστήμη οι μηχανικοί έχουν βρει τρόπο για την αντιμετώπιση αυτού του προβλήματος.

2.1.3.2 Ιατρική

Στην παρακάτω φωτογραφία φαίνεται η εξομοίωση μιας οδοντικής χειρουργικής επέμβασης που περιλαμβάνει τα στάδια της ανίχνευσης, της ανίχνευσης περιοδοντικού βάθους και της επέμβασης. Ο στόχος της προσομοίωσης είναι η εξομοίωση της φυσικής επαφής μεταξύ του οδοντικού εργαλείου και του δοντιού, της γλώσσας ή των ούλων ανάλογα επίσης και με τα

επίπεδα φθοράς των δοντιών. Η σκληρότητα και η τριβή του κάθε ιστού μπορούνε να προσομοιωθούν και να παρέχουν ανάδραση στην απτική συσκευή του χρήστη [B2].



Εικόνα 7: Απτική τεχνολογία στην οδοντιατρική

(Πηγή: https://www.researchgate.net/figure/Dental-surgical-simulator-with-bi-manual-haptic-feedback-a-Visio-haptic-collocated_fig6_332510833)

2.1.3.3 Βιντεοπαιχνίδια και εικονική πραγματικότητα

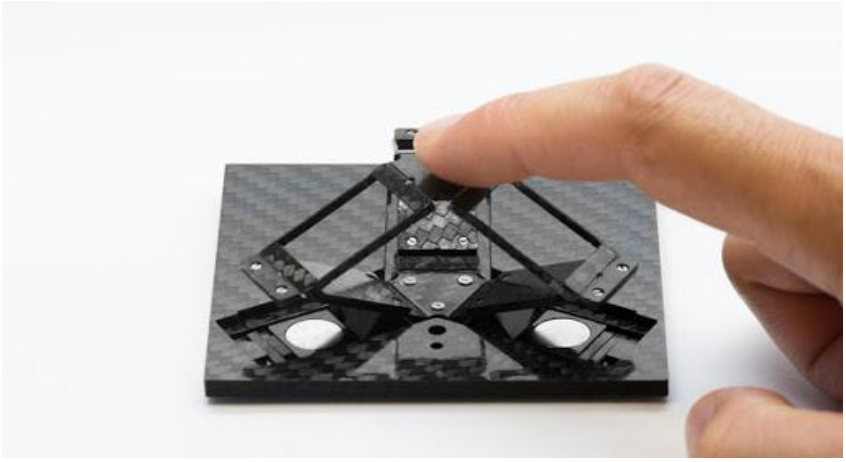
Οι απτικές συσκευές κερδίζουν ολοένα και περισσότερο έδαφος στην εικονική πραγματικότητα καθώς προσθέτουν την αίσθηση της αφής πέρα από την οπτική αίσθηση [B9]. Αρκετά συστήματα αναπτύσσονται για να κάνουν χρήση απτικών διεπαφών για 3D σχεδίαση και μοντελοποίηση ενώ θα επιτρέπουν τη δυνατότητα ο χρήστης να βλέπει και να αισθάνεται ακόμη και ολογράμματα. Ένας σημαντικός αριθμός εταιριών αναπτύσσουν απτικά γιλέκα ή ακόμη και ολόκληρες στολές για μια πλήρη και σε βάθος αναβίωση μιας εικονικής πραγματικότητας.

2.1.4 State of the art

Παρακάτω παρατίθενται δυο αναφορικά παραδείγματα από το χώρο των απτικών συσκευών που μελετούνται από πανεπιστήμια και ερευνητικά κέντρα ανά τον κόσμο.

2.1.4.1 Foldaway haptic

Το εργαστήριο του Jamie Paik στο Ελβετικό Ομοσπονδιακό Ινστιτούτο Τεχνολογίας στη Λωζάννη (EPFL) έχει κατασκευάσει μια φορητή απτική συσκευή που ονομάζεται Foldaway. Περιλαμβάνει 3 βραχίονες που ενώνονται στο ανώτερο σημείο όπου ο χρήστης τοποθετεί το δάχτυλό του χρησιμοποιώντας το ως χειριστήριο (joystick) σε τρισδιάστατο επίπεδο και παρέχει την κατάλληλη αντίσταση ως ανάδραση.

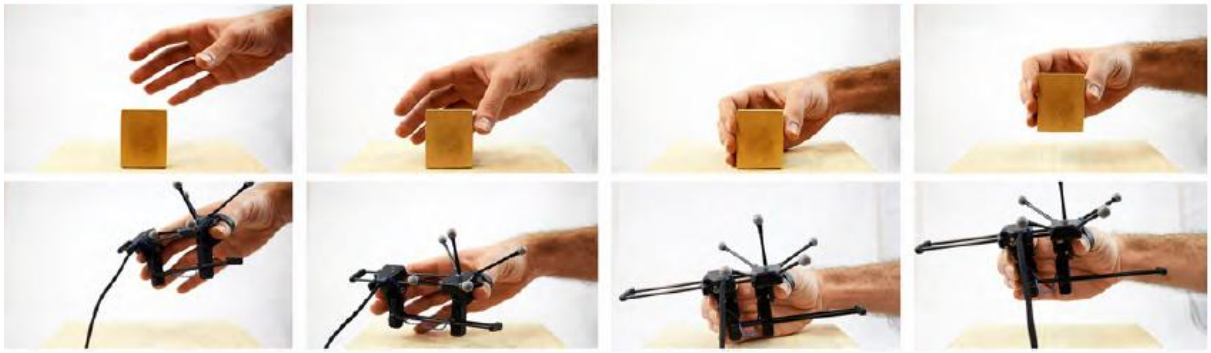


Εικόνα 8: Η Foldaway haptic απτική συσκευή

(Πηγή: https://www.eenewseurope.com/news/origami-unfolds-force-feedback-haptics)

2.1.4.2 Grabity

Ο Culbertson και η ομάδα του έχουν αναπτύξει μια φορητή συσκευή που ονομάζουν **Grabity** και προσομοιώνει την ψευδαίσθηση του βάρους και της αδράνειας ανάλογα με τον τρόπο που δονείται [B6].

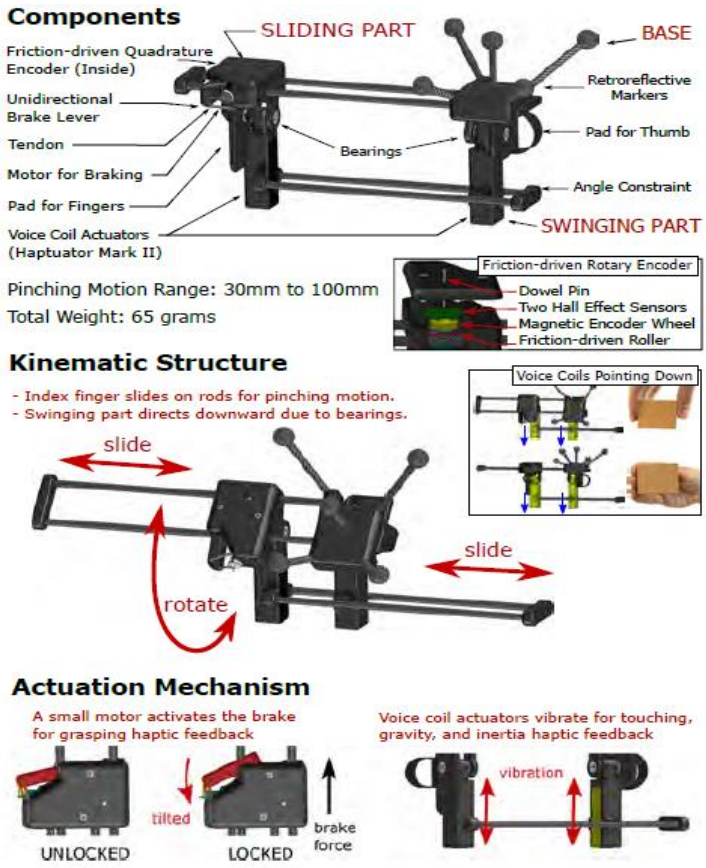


Εικόνα 9: Η απτική συσκευή Grabity

(Πηγή: http://shape.stanford.edu/research/grabity/)

Η συσκευή αποτελείται από 3 μέρη: τη βάση, το συρόμενο μέρος και το κινούμενο μέρος. Η βάση στηρίζεται στον αντίχειρα και διαθέτει ανακλαστήρες για την ανίχνευση από της θέσης και του προσανατολισμού του αντίχειρα. Το κυλιόμενο μέρος είναι τοποθετημένο στον δείκτη και συνδέεται με τη βάση μέσω μιας πρισματικής άρθρωσης. Αυτή η άρθρωση επιτρέπει στον χρήστη να αγγίξει ένα αντικείμενο με τα 2 δάχτυλα (σαν αρπάγη). Διαθέτει επίσης και έναν μηχανισμό φρένου. Το κινούμενο μέρος το οποίο συνδέεται με το κυλιόμενο μέρος και τη βάση μέσω περιστροφικών συνδέσμων αποτελείται από δυο ευθύγραμμους ενεργοποιητές (voice coil actuators) και μιας πρισματικής άρθρωσης. Κάθε ευθύγραμμος ενεργοποιητής τοποθετείται κοντά μεταξύ του δείκτη και του αντίχειρα ώστε να μεταδίδει τα επιθυμητά σήματα δόνησης. Το βάρος της κατασκευής αγγίζει τα 65 γραμμάρια.

Ο μηχανισμός φρένου του Grability χρησιμοποιείται για τη δημιουργία μιας δύναμης πιασίματος, ενώ οι ενεργοποιητές παρέχουν αίσθηση αφής κατά την αρχική επαφή ενώ παρέχουν επίσης και την αίσθηση του βάρους όταν ο χρήστης σηκώνει ένα αντικείμενο.



Εικόνα 10: Μηχανολογική κατασκευή της απτικής συσκευής Grability

(Πηγή: http://shape.stanford.edu/research/grabity/)

Όταν ο χρήστης σηκώνει ένα εικονικό αντικείμενο, οι ενεργοποιητές δονούνται ασυμμετρικά για να εξομοιώσουν την αίσθηση του βάρους. Με την εναλλαγή της ασυμμετρίας των δονήσεων η συσκευή εξομοιώνει τα διάφορα βάρη ενός αντικειμένου. Ένας μικροελεγκτής στέλνει κατάλληλους παλμούς ρεύματος στους ενεργοποιητές που προκαλούνε μια μεγάλη επιτάχυνση στον μαγνήτη του και στη συνέχεια τον επιβραδύνει σιγά σιγά στην αρχική του θέση.

2.2 Η μηχανική όραση

Η χρήση της μηχανικής όρασης κυρίως στο κομμάτι της παραγωγής και αυτοματοποίησης αναμένεται να αυξηθεί σημαντικά καθώς μπορούνε να χρησιμοποιηθούνε παράλληλα με αλγορίθμους βαθιάς εκμάθησης (deep learning) που βελτιστοποιούν τα επίπεδα της παραγωγής [Δ7].

Εικόνα 11: Βιομηχανική κάμερα μηχανικής όρασης

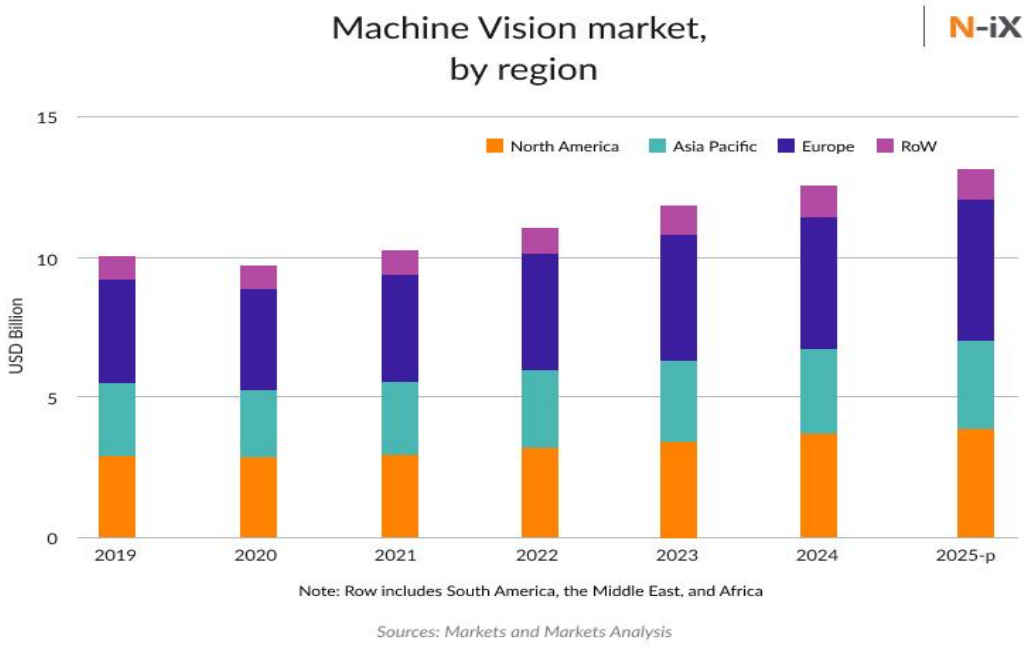
(Πηγή: https://www.roboticstomorrow.com/article/2020/03/cut-from-the-same-cloth-automatic-3d-recognition-and-marking-of-wooden-beams/15050/)

Με την είσοδο μας στο μοντέλο του Industry 4.0 όπου γίνεται η ψηφιοποίηση όλων των παραγωγικών δραστηριοτήτων, αρκετοί επιστήμονες και ερευνητές μελετούν διαρκώς νέους τρόπους και μεθόδους για την βελτιστοποίηση αυτών των δραστηριοτήτων. Ο στόχος είναι η

δημιουργία έξυπνων μηχανών που θα βλέπουν, θα επικοινωνούν και θα κάνουν μια εργασία με μεγαλύτερη ακρίβεια και αξιοπιστία.

Η μηχανική όραση αποτελεί μέρος ενός έξυπνου εργοστασίου καθώς επιτρέπει σε μια μηχανή να ‘’βλέπει’’ τον φυσικό κόσμο και να επεξεργάζεται και να αναλύει πληροφορίες βασισμένες σε οπτικές εισόδους.

Από το 2018, η παγκόσμια αγορά για τη μηχανική όραση άγγιζε τα 9.2 δις δολλάρια και αναμένεται να ξεπεράσει τα 13 δις δολλάρια ως το τέλος το 2025.



Γράφημα 1: Η πρόβλεψη της χρήσης της μηχανικής όρασης στην παγκόσμια αγορά

(Πηγή: https://www.n-ix.com/computer-vision-manufacturing)

Παράλληλα με την ανάπτυξη της τεχνητής νοημοσύνης ο συνδιασμός της με τη μηχανική όραση ανοίγει νέους ορίζοντες στην παραγωγική διαδικασία. Τα οφέλη των 2 αυτών τεχνολογιών συνοψίζονται παρακάτω:

- Μειωμένο κόστος παραγωγής και μείωση των ελαττωματικών προϊόντων.

- Αυξημένη ακρίβεια και επαναληψιμότητα.

- Αυξημένα επίπεδα ασφάλειας του προσωπικού.

Η μηχανική όραση συναντάται στις ακόλουθες παραγωγικές διεργασίες σε μια βιομηχανία. Παρακάτω παρατίθενται τα πιο διαδεδομένα εργαλεία και βιβλιοθήκες που χρησιμοποιούνται για την μηχανική όραση.

OpenCV: Είναι το πιο διαδεδομένο εργαλείο για την μηχανική όραση. Είναι γραμμένη σε C++ και διαθέτει bindings για python, java, matlab και javascript. Διαθέτει πληθώρα αλγορίθμων και συναρτήσεων για μηχανική όραση, επεξεργασία εικόνων, ακόμη και 3d ανακατασκευή.

VisionWorks: Ένα ακόμη αρκετά διαδεδομένο εργαλείο, από την Nvidia και χρησιμοποιείται σε εφαρμογές όπως ανάλυση video, αλγορίθμους εντοπισμού (Localization) καθώς προσφέρει ευκολία και ταχύτητα για την ανίχνευση και ανάλυση πολλαπλών σκηνών.

AForge.NET/Accord.NET και **Computer Vision Sandbox**. Χρησιμοποιείται κυρίως για .Net εφαρμογές. Το **Computer Vision Sandbox** είναι ανοιχτού κώδικα και επιτρέπει στο χρήστη να επιλύσει προβλήματα όπως αυτοματισμούς βασιζόμενους σε μηχανική όραση, επεξεργασία εικόνας και βίντεο, ανίχνευση barcode κ.α.

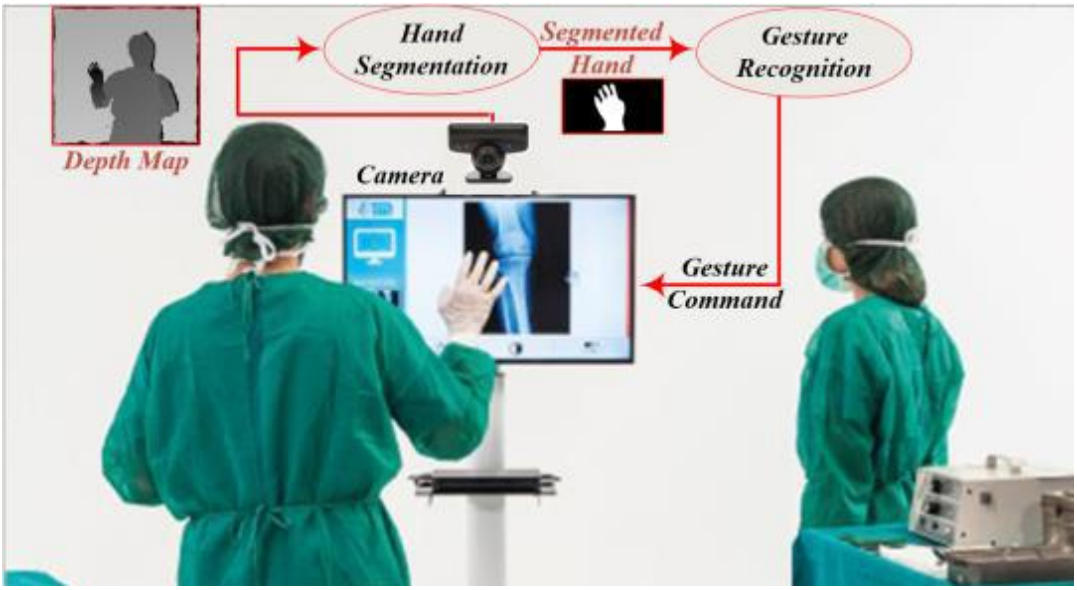
SimpleCV είναι framework για την επίλυση προβλημάτων μηχανικής όρασης για πιο απλές εφαρμογές μηχανικής όρασης από τα προαναφερθέντα, και θα μπορούσε να χαρακτηριστεί ως μια ελαφριά και λιγότερων δυνατοτήτων OpenCV. Θα πρέπει να σημειωθεί επίσης υπάρχει σαφώς μικρότερη κοινότητα συντήρησης και ανάπτυξης της simplecv από ότι της OpenCV.

VisionWorkbench η οποία είναι μια βιβλιοθήκη της ΝΑΣΑ και είναι γραμμένη σε C++. Χρησιμοποιείται κυρίως για ανάλυση και ενίσχυση εικόνας.

PyTorch η οποία είναι μια ανοιχτού κώδικα βιβλιοθήκη μηχανικής όρασης η οποία αναπτύχθηκε κυρίως από το τμήμα έρευνας του Facebook. Διαθέτει διεπαφή (Interface) για python και C++.

2.2.1 Μηχανική όραση στην Ιατρική

Στον τομέα της ιατρικής είναι ιδιαίτερα σημαντικό στους χειρουργικούς θαλάμους οι χειρούργοι ιατροί να μπορούνε να χρησιμοποιούνε τα χέρια τους χωρίς να αγγίζουν περισσότερα αντικείμενα για να διατηρούνε τα χέρια τους αποστειρωμένα. Ένα τέτοιο παράδειγμα αποτελεί το Gestix το οποίο είναι ένα ιατρικό εργαλείο με μια μεγάλη οθόνη όπου προβάλλει σχετική πληροφόρηση στους ιατρούς. Δέχεται εντολές από συγκεκριμένες χειρονομίες του χεριού ενώ παράλληλα δίνει τα κατάλληλα εργαλεία στον χειρούργο ιατρό.



Εικόνα 12: Ανίχνευση χειρονομιών σε χειρουργικό θάλαμο

(Πηγή: arxiv.org [B2])

Η αναγνώριση χειρονομιών αποτελείται από 3 στάδια: την ανίχνευση, την παρακολούθηση και την αναγνώριση. Συνήθη χαρακτηριστικά που εξάγονται είναι οι γωνίες, οι αποστάσεις, οι θέσεις των κορυφών των δαχτύλων, το κέντρο της παλάμης, ο προσανατολισμός του χεριού

και άλλα. Έπειτα τα χαρακτηριστικά αυτά τροφοδοτούνται σε ένα νευρωνικό δίκτυο για ταξινόμηση (classification) και τελική αναγνώριση. Λόγω του μεγάλου όγκου επεξεργασίας προτείνεται η χρήση κάρτας γραφικών ή ενός FPGA.

Ένα από τα κύρια προβλήματα είναι η απόκριση σε πραγματικό χρόνο και για αυτό το λόγο πολλοί ερευνητές και πανεπιστήμια μελετούνε συνεχώς νέους πιο γρήγορους και αποδοτικούς αλγορίθμους για την επίλυση παρόμοιων προβλημάτων.

Οι μέθοδοι επίλυσης με χρήση βαθιάς μάθησης (deep learning) είναι κατάλληλες για προβλήματα αναγνώρισης μοτίβων και συγκεκριμένα τα συνελικτικά νευρωνικά δίκτυα είναι από τις πιο αποδοτικές μεθόδους βαθιάς μάθησης για εξαγωγή χαρακτηριστικών από μια μεγάλη ομάδα εικόνων.

Κεφάλαιο 3

3.1 Εισαγωγή – Ερευνητική μέθοδος

Στο κομμάτι της κατασκευής συναντήσαμε αρκετούς προβληματισμούς και χρησιμοποιήθηκαν συγκεκριμένες μεθοδολογίες για την επίλυση αυτών όπως για παράδειγμα η χρήση του φίλτρου Kalman στους αισθητήρες κάμψης. Επιπλέον αναπτύχθηκε αλγόριθμος ώστε τα πακέτα μέσω της ασύρματης Bluetooth επικοινωνίας που λαμβάνει ο receiver, να τα διαβάζει ένας δεύτερος Arduino μικροελεγκτής ώστε ακόμη και εάν αποσυγχρονιστούνε, να λαμβάνονται με τη σωστή σειρά ανά πάσα στιγμή και να μην κινούνται λανθασμένα οι κινητήρες του ρομποτικού χεριού. Στο mode ελέγχου μέσω κάμερας χρησιμοποιήθηκε η βιβλιοθήκη OpenCV η οποία διευκολύνει κατά πολύ τον χρήστη στην δημιουργία αλγορίθμων για μηχανική όραση. Επιπροσθέτως, στο τέλος του κεφαλαίου αυτού παρατίθεται και ένας πίνακας με τη λίστα και το ενδεικτικό κόστος των υλικών που χρησιμοποιηθήκανε.

3.2 Το Kalman φίλτρο

Το φίλτρο Kalman είναι ιδιαίτερα χρήσιμο φίλτρο καθώς δεν ξοδεύει σημαντικούς υπολογιστικούς πόρους ενός συστήματος, είναι γρήγορο και αποτελεσματικό, χαρακτηριστικά ιδιαίτερα χρήσιμα σε μικρά ενσωματωμένα συστήματα και μικροελεγκτές. Επιπλέον, είναι ιδανικό για συστήματα με συχνές μεταβολές της εξόδου ή εξόδων του.

Ας εξετάσουμε όμως το παρακάτω ενδεχόμενο. Κάθε αισθητήρας (encoder, GPS, αναλογικά αισθητήρια κτλ.) μας δίνει την πληροφορία ή πληροφορίες που χρειαζόμαστε, με μια αρκετά καλή ακρίβεια ανάλογα και με την ανάλυση που μας παρέχει. Στην πραγματικότητα όμως, σε ένα σύστημα υπεισέρχονται και αστάθμητοι παράγοντες όπως π.χ σε ένα αυτόνομο όχημα που κινείται στο χώρο τα encoder μπορεί να μην μας δίνουν ακριβή μέτρηση καθώς μπορεί μια ή και περισσότερες ρόδες να ολισθήσουν με συνέπεια η μετρούμενη τιμή για την εύρεση της ταχύτητας και της θέσης να μην είναι και τόσο ακριβής πλέον καθώς και ένα σφάλμα από τους ίδιους τους αισθητήρες.

Σε αυτό το σημείο έρχεται να μας βοηθήσει το φίλτρο Kalman. Αναπτύχθηκε από τον Rudolf Kalman και ονομάζεται και ‘’γραμμική τετραγωνική εκτίμηση’’ (linear quadratic estimation). Το Kalman φίλτρο δέχεται ως είσοδο την πληροφορία η οποία περιέχει ένα ποσοστό λάθους, και μειώνει τον θόρυβο ή την αβεβαιότητα της πληροφορίας αυτής. Ένα ακόμη χαρακτηριστικό γνώρισμα του φίλτρου είναι ότι περιγράφει συστήματα γραμμικού χώρου – χρόνου.

Το Kalman φίλτρο χρησιμοποιεί μια πρόβλεψη που ακολουθείται από μια διόρθωση ώστε να ορίσει τις καταστάσεις του φίλτρου. Η κύρια ιδέα είναι ότι χρησιμοποιώντας πληροφορίες σχετικές με την δυναμική μιας κατάστασης, τότε το φίλτρο θα προβλέψει ποια θα είναι η επόμενη κατάσταση του συστήματος.

Ξεκινώντας από μια αρχική κατάσταση εκτίμησης, x_0, με έναν αρχικό πίνακα σφάλματος κατάστασης συνδιακύμανσης (covariance matrix) P_0, εφαρμόζεται το σύστημα πρόβλεψης - διόρθωσης επαναληπτικά [B10]. Αρχικά προβλέπεται το διάνυσμα κατάστασης από την γενική εξίσωση της δυναμικής της κατάστασης

$$\mathbf{x}_{k|k-1} = F_{k-1}x_{k-1}+G_{k-1}u_{k-1} \qquad (3)$$

Όπου $x_{k|k-1}$ είναι το διάνυσμα κατάστασης που προβλέφθηκε, x_{k-1} είναι το προηγούμενο διάνυσμα κατάστασης που προβλέφθηκε, u είναι το διάνυσμα εισόδων, ενώ F και G είναι πίνακες που ορίζουν τη δυναμική (κατάσταση) του συστήματος. Στη συνέχεια πρέπει προβλεφθεί ο πίνακας συνδιακύμανσης της κατάστασης από την ακόλουθη εξίσωση:

$$P_{k|k-1} = F_{k-1}P_{k-1}F^{T}_{k-1}+Q_{k-1} \qquad (4)$$

Όπου ο όρος $P_{k|k-1}$ αντιπροσωπεύει τον προβλεφθέντα πίνακα συνδιακύμανσης της κατάστασης, P_{k-1} είναι η προηγούμενη εκτίμηση για τον πίνακα συνδιακύμανσης και το Q είναι ο πίνακας συνδιακύμανσης του θορύβου του συστήματος.

Στη συνέχεια μπορούμε να υπολογίσουμε το Kalman κέρδος (Kalman Gain) με την εξίσωση:

$$K_k = P_{k|k-1}H_k^{T}(H_kP_{k|k-1}H_k^{T}+R_k)^{-1} \quad (5)$$

Ο πίνακας H είναι απαραίτητος για τον καθορισμό της εξόδου, και R είναι ένας πίνακας της συνδιακύμανσης του θορύβου της μέτρησης. Το διάνυσμα της κατάστασης ανανεώνεται με την διαφορά μεταξύ της μέτρησης της εξόδου και της προβλεπόμενης εξόδου, πολλαπλασιασμένο με τον πίνακα του κέρδους Kalman για να διορθώσει την πρόβλεψη.

$$x_k = x_{k|k-1} + K_k(z_k - H_k x_{k|k-1}) \qquad (6)$$

Παρομοίως ανανεώνεται και ο πίνακας συνδιακύμανσης της κατάστασης του σφάλματος.

$$P_k=(I - K_k H_k)P_{k|k-1}. \qquad (7)$$

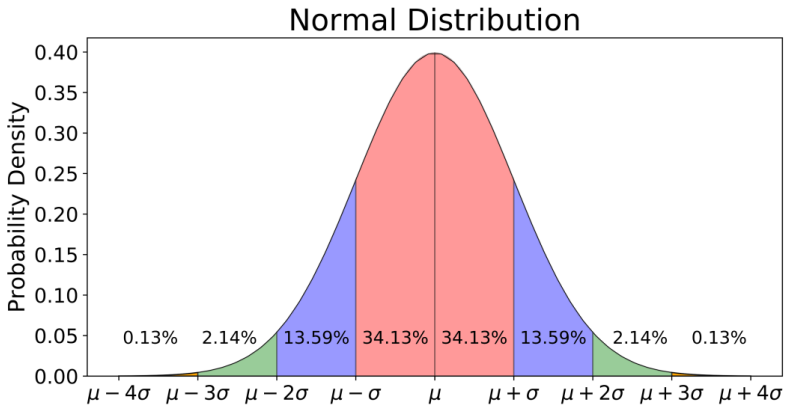
Όπου ο I είναι ένας μοναδιαίος πίνακας.

3.3 Υλοποίηση 1d Kalman φίλτρου

Στη συγκεκριμένη περίπτωση θα εξετάσουμε το μονοδιάστατο (1d) Kalman φίλτρο [Δ9] καθώς έχουμε μόνο μια μεταβλητή που μετράμε (μονοδιάστατο πρόβλημα) η οποία είναι η αναλογική τιμή των αισθητήρων κάμψης και δεν συσχετίζεται με άλλη μεταβλητή.

Η γενική λειτουργία του φίλτρου αυτού είναι η εξής:

Σε ένα σύστημα με διαρκώς μεταβαλλόμενη έξοδο, μια μετρούμενη μεταβλητή είναι τυχαία αλλά ακολουθεί μια Gaussian κατανομή.



Γράφημα 2: Gaussian κατανομή

(Πηγή: https://towardsdatascience.com/understanding-the-68-95-99-7-rule-for-a-normal-distribution-b7b7cbf760c2)

Κάθε τιμή έχει έναν μέσο όρο που ορίζεται ως το κεντρικό σημείο της ανωτέρω κατανομής (μ) καθώς και την τυπική απόκλιση (σ) που ερμηνεύεται και ως η αβεβαιότητα μιας μετρούμενης μεταβλητής.

Kalman κέρδος (Kalman Gain)

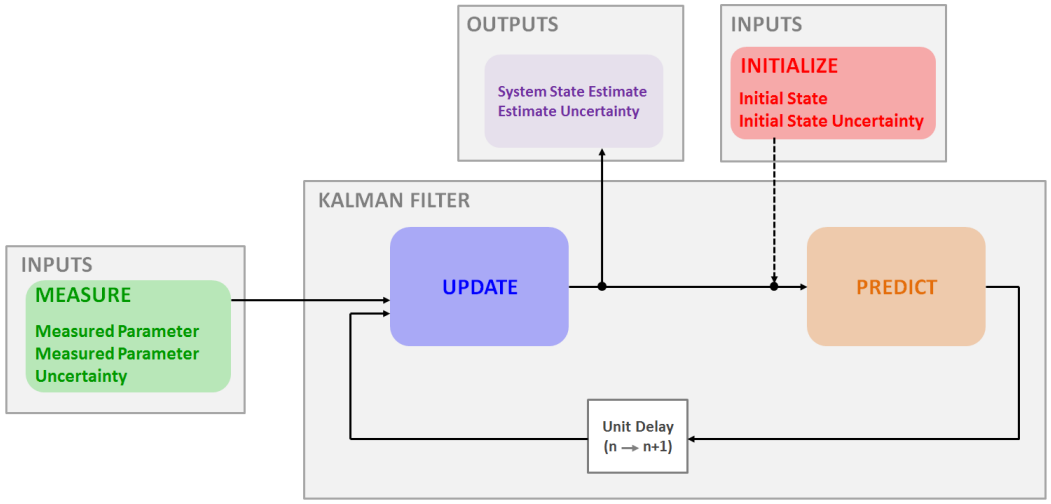
Το κέρδος Kalman έχει την εξής μορφή:

$$KG = \frac{\alpha\beta\varepsilon\beta\alpha\iota\acute{o}\tau\eta\tau\alpha \quad \sigma\tau\eta\nu \quad \tau\omega\rho\iota\nu\acute{\eta} \quad \varepsilon\kappa\acute{\tau}\iota\mu\eta\sigma\eta}{\alpha\beta\varepsilon\beta\alpha\iota\acute{o}\tau\eta\tau\alpha \quad \sigma\tau\eta\nu \quad \tau\omega\rho\iota\nu\acute{\eta} \quad \varepsilon\kappa\acute{\tau}\iota\mu\eta\sigma\eta + \alpha\beta\varepsilon\beta\alpha\iota\acute{o}\tau\eta\tau\alpha \quad \sigma\tau\eta\nu \quad \mu\acute{\varepsilon}\tau\rho\eta\sigma\eta} = \frac{pn}{pn + rn}$$

(1α) και παίρνει τιμές από 0 έως 1. Πρακτικά το κέρδος Kalman μας λέει πόσο θέλουμε να αλλάξουμε την εκτίμηση της τιμής με δεδομένο την μετρούμενη τιμή.

Όταν το κέρδος Kalman είναι κοντά στο 0, σημαίνει ότι η αβεβαιότητα της εκτίμησης είναι πολύ μικρή ενώ η αβεβαιότητα της μέτρησης είναι υψηλή που σημαίνει ότι δίνουμε μεγάλη βαρύτητα στην εκτίμηση της τιμής που μετράμε παρά στις τιμές που παίρνουμε από τα όργανα (ή αισθητήρα) μέτρησης. Όταν το κέρδος αυτό έχει τιμή 1, αυτό σημαίνει ότι δίνουμε αισθητά μεγαλύτερη βαρύτητα στις μετρούμενες τιμές που παίρνουμε από τα αισθητήρια, παρά από τις εκτιμήσεις των μετρούμενων τιμών. Όταν το κέρδος έχει τιμή γύρω στο 0,5 σημαίνει ότι δίνουμε την ίδια βαρύτητα για να προκύψει η τελική τιμή από την αβεβαιότητα στην εκτίμηση και την αβεβαιότητα στη μέτρηση. Όσο παίρνουμε περισσότερες μετρήσεις, το κέρδος αυτό τείνει στο μηδέν.

Το Kalman φίλτρο ουσιαστικά αποτελείται από τα βήματα ‘’μετράω’’ (measure), “ανανεώνω’’ (update), ‘’εκτιμώ’’ (predict) που συνοψίζεται στο παρακάτω σχήμα.



Γράφημα 3: Σχηματική απεικόνιση μονοδιάστατου Kalman φίλτρου

(Πηγή: https://www.kalmanfilter.net/kalman1d.html)

Λειτουργία 1d (μονοδιάστατου) Kalman φίλτρου

Βήμα 0 -Αρχικοποίηση

Ως πρώτο βήμα θεωρείται η αρχικοποίηση που περιλαμβάνει την εκτίμησή μας για την τιμή που θα μετρήσουμε στις επαναλήψεις, και δίνουμε τιμή στην αβεβαιότητα της εκτίμησης (estimate uncertainty).

1η επανάληψη

Στην πρώτη επανάληψη, παίρνουμε την πρώτη μέτρηση από τους αισθητήρες κάμψης. Από την τυπική απόκλιση (standard deviation σ) του σφάλματος μέτρησης (από datasheets αισθητηρίων, οργάνων μέτρησης κτλ) προκύπτει και η διακύμανση (σ^2) που είναι και η αβεβαιότητα της μέτρησης (measurement uncertainty).

Α) Υπολογίζουμε το κέρδος Kalman από τον τύπο 1α.

Β) Υπολογίζουμε επίσης τη νέα εκτίμηση (current estimate) από τον τύπο:

Νέα εκτίμηση = Παλιά εκτίμηση + Κέρδος Kalman * (μετρηθείσα τιμή – προηγούμενη εκτίμηση) δηλαδή συνοψίζεται ως

$X_n = X_{n-1} + K_G * (M - X_{n-1})$ (2).

Υπολογίζουμε την αβεβαιότητα της τωρινής εκτίμησης με τον τύπο:

Αβεβαιότητα τωρινής εκτίμησης = (1 - Κέρδος Kalman) * Αβεβαιότητα στην τωρινή εκτίμηση

Ue = (1- K_G) * pn (3).

Γ) Ανανέωση τιμών όπου η τιμή αυτή χρησιμοποιείται ως αρχική στην επόμενη επανάληψη.

Ομοίως γίνονται όλες οι επόμενες επαναλήψεις καθώς το Kalman φίλτρο βελτιώνεται με συνέπεια η αβεβαιότητα της εκτίμησης να μικραίνει διαρκώς.

3.4 Η Βιβλιοθήκη OpenCV

Εικόνα 13: Λογότυπο της OpenCV

(Πηγή: https://en.wikipedia.org/wiki/OpenCV)

Η OpenCV (Open Source Computer Vision Library) είναι λογισμικό ανοιχτού κώδικα για μηχανική όραση και χρησιμοποιείται ευρέως σε μηχανική μάθηση (machine learning) και νευρωνικά δίκτυα (neural networks) [Δ8]. Η OpenCV φτιάχτηκε για να διευκολύνει τον μέσο χρήστη καθώς και επιχειρήσεις σε εφαρμογές όπου η χρήση κάμερας είναι απαραίτητη. Διαθέτοντας BSD άδεια χρήσης δεν προβάλει φραγμούς στη χρήση της από οποιονδήποτε χρήστη.

Διαθέτει επίσης περισσότερους από 2500 βελτιστοποιημένους αλγορίθμους για ανίχνευση, αναγνώριση αντικειμένων ή προσώπων, αναγνώριση κινούμενων αντικειμένων ή προσώπων, στη ρομποτική, στην παραγωγή 3D μοντέλων, σε μοντέλα προεκπαιδευμένων νευρωνικών

δικτύων καθώς πληθώρα άλλων λειτουργιών. Υποστηρίζει όλα τα γνωστά λειτουργικά συστήματα όπως Windows, Linux, Mac-OS και Android. Η OpenCV είναι γραμμένη σε C++ αλλά μέσω των API's ένας χρήστης μπορεί να γράψει τον κώδικά του σε Python, Java, Matlab, C++.

Η OpenCV έχει περισσότερους από 47.000 χρήστες της κοινότητας (user community) ενώ χρησιμοποιείται και από εταιρίες διεθνούς εμβέλειας όπως είναι η Google, Yahoo, Microsoft, Intel, IBM, Sony καθώς και από μικρές startup εταιρίες. Μπορεί να χρησιμοποιηθεί σε περιπτώσεις όπου χρειάζεται να ενώσουμε πολλές φωτογραφίες σε μία (Google street view), ανίχνευση εισβολέων σε έναν χώρο, παρακολούθηση ανθρώπων σε χώρους όπως πισίνες για αποφυγή πνιγμών, ανίχνευση εμποδίων σε οδικά δίκτυα, ποιοτική ανίχνευση προϊόντων, αναγνώριση προσώπων και άλλες. Η OpenCV μπορεί να συνεργαστεί εύκολα με frameworks για βαθιά μάθηση (Deep Learning) και μοντέλα νευρωνικών δικτύων όπως το Tensorflow, το Torch/PyTorch και το Caffe.

Μερικές από τις εφαρμογές της OpenCV μπορούν να συνοψιστούνε παρακάτω:

- Αναγνώριση προσώπου
- Συστήματα ασφαλείας
- number of people – count (foot traffic in a mall, etc)
- Αναγνώριση αριθμού αυτοκινήτων και τις ταχύτητές τους.
- Ανίχνευση ανωμαλιών και ελαττωματικών προϊόντων στη βιομηχανία.
- Σύνθεση μικρότερων εικόνων σε μια μεγαλύτερη
- Επεξεργασία εικόνων και video
- Αυτόνομα οχήματα
- Αναγνώριση αντικειμένων
- Ανάλυση ιατρικών εικόνων
- Ταινίες

Στην επίσημη ιστοσελίδα της OpenCV μπορούμε να βρούμε οδηγίες (tutorials) ανάλογα με την αντίστοιχη εφαρμογή όπως για παράδειγμα:

- Βασικές λειτουργίες της OpenCV όπως διάβασμα εικόνων, επεξεργασία, αλλαγή μεγέθους, μετατροπή σε Grayscale, φιλτράρισμα κ.α

- Ανάλυση βίντεο και μεθόδους όπως η αφαίρεση του φόντου (background subtraction), mean shift και cam shift

- Περιγραφείς αναγνώρισης όπως οι αλγόριθμοι SIFT, SURF, BRIEF, FAST καθώς και αλγορίθμους ανίχνευσης γωνιών Harris, Shi Tomasi κ.α

- Λειτουργίες για κάμερες όπως το καλιμπράρισμα, η επιπολική γεωμετρία, ο χάρτης βάθους (depth map) κ.α

- Ανίχνευση αντικειμένων με ταξινομητές αλληλουχίας (cascade classifier)

- Μηχανική μάθηση με αλγορίθμους όπως οι K-Nearest Neighbour, Support Vector Machines (SVM), K-Means Clustering

Σημείωση: Στην υλοποίηση του κώδικα χρησιμοποιήθηκε Python. Ως γνωστόν, η Python, ως μεταφραζόμενη (interpreted) γλώσσα, είναι σημαντικά πιο αργή από compiled γλώσσες όπως η C ή η C++ κάτι που είναι πολύ σημαντικό σε αυτόνομα συστήματα όπου η ταχύτητα εκτέλεσης ενός προγράμματος είναι καθοριστικός παράγοντας. Στην περίπτωση όμως της OpenCV, δεν τρέχει Python στην ουσία, αλλά χρησιμοποιείται ως wrapper (περιτύλιξη για Python) με συνέπεια η διαφορά στην ταχύτητα εκτέλεσης ενός προγράμματος, είτε γράφουμε σε C++ είτε σε Python, να είναι σχεδόν αμελητέα. Η χρήση της Python συστήνεται λόγω μεγαλύτερης ευκολίας για την υλοποίηση πολύπλοκων αλγορίθμων ενώ επίσης προκύπτουν σημαντικά λιγότερες γραμμές κώδικα για την ίδια υλοποίηση.

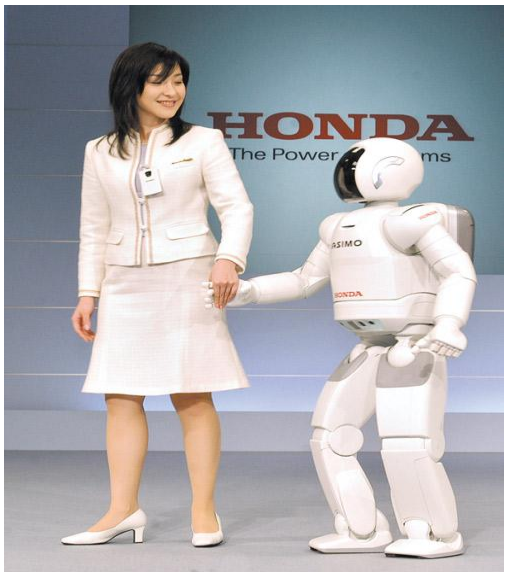
3.5 Ρομποτικό χέρι

Ένα ρομποτικό χέρι είναι συνήθως ένα μηχανικό, προγραμματιζόμενο χέρι με λειτουργίες που θυμίζουνε αυτές του ανθρώπινου. Υπάρχουν σύνδεσμοι (φάλαγγες) που ενώνονται με αρθρώσεις για να προκύψει η τελική κίνηση. Ρομποτικά χέρια χρησιμοποιούνται στην

ιατρική (ως αντικατάσταση χαμένων μελών), στην ψυχαγωγία (ταινίες), στη βιομηχανία και σε ανθρωπόμορφα ρομπότ.

Εικόνα 14: Προσθετικό χέρι της Deka

(Πηγή: https://www.businessinsider.com/deka-arm-mind-controlled-prosthetic-limb-2014-5)



Εικόνα 15: Το ρομπότ της Honda Asimo

(Πηγή: https://www.taiwannews.com.tw/en/news/137304)

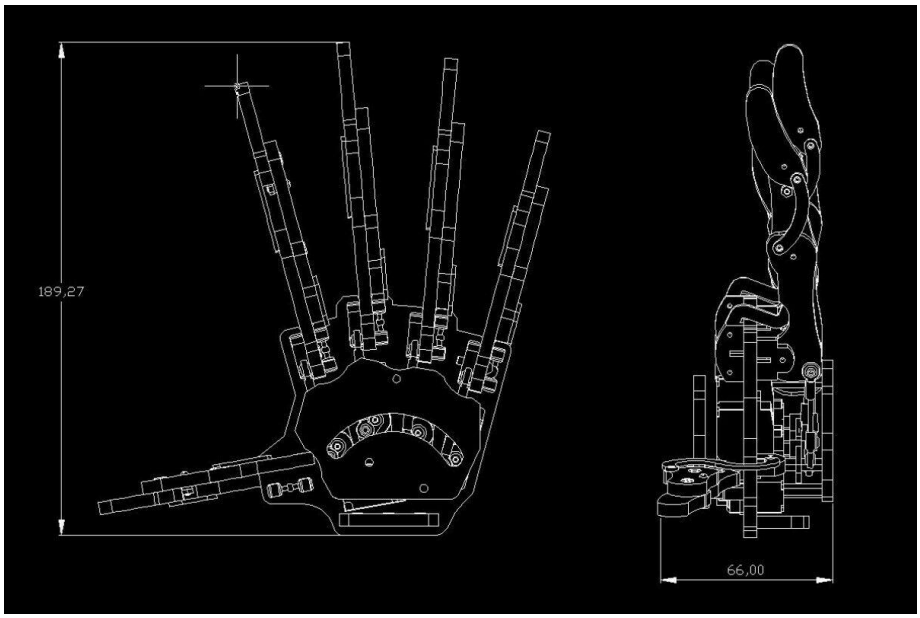
Τα τελευταία χρόνια υπάρχει μεγάλος αριθμός εταιριών που ασχολούνται με τη δημιουργία τεχνητών προσθετικών μελών για καλύτερο έλεγχο, αξιοπιστία και εργονομία για τους χρήστες. Συνηθισμένα υλικά κατασκευής είναι το αλουμίνιο, τα ανθρακονήματα, τιτάνιο και άλλα κράματα για αυξημένη μηχανική αντοχή και μικρό βάρος. Με τη χρήση εξελιγμένων αισθητήρων τα ρομποτικά χέρια είναι σε θέση να εκτελούνε πληθώρα λειτουργιών και εφαρμόζοντας την κατάλληλη πίεση ανάλογα με το αντικείμενο που είναι να συγκρατήσουν.

Το ρομποτικό χέρι που χρησιμοποιήθηκε δεν χρειάστηκε σχεδίαση ή τοποθέτηση των επιμέρους εξαρτημάτων. Διαθέτει 5 αρθρώσεις στο σύνολο, δηλαδή αποτελείται από 5 σερβοκινητήρες, έναν για κάθε δάχτυλο. Η κίνηση των δαχτύλων γίνεται μέσω ενός μπράτσου που κάνει έκταση και σύμπτυξη στον σύνδεσμο που βρίσκεται στην αρχή του κάθε δαχτύλου. Κάθε δάχτυλο αποτελείται από 3 επιμέρους κομμάτια που συνδέονται και αυτά μεταξύ τους με κυρτούς μεταλλικούς συνδέσμους.

Εικόνα 16: Ρομποτικό χέρι 5 βαθμών ελευθερίας

(Σύνδεσμος: ebay.com)

Το ανθρώπινο χέρι έχει 27 βαθμούς ελευθερίας που σημαίνει ότι η σχεδίαση, η τοποθέτηση και η ακρίβεια ενός ρομποτικού χεριού που μιμείται ένα ανθρώπινο, μπορεί να γίνει ένα ιδιαίτερα απαιτητικό στη λύση πρόβλημα. Στη συγκεκριμένη εφαρμογή δεν εστιάσαμε στην ακρίβεια της κίνησης παρά μόνο στην ασύρματη μετάδοση των σημάτων των αισθητήρων και στον έλεγχο από κάμερα μέσω της OpenCV.



Εικόνα 17: Μηχανολογικό σχέδιο ρομποτικού χεριού

(Σύνδεσμος: ebay.com)

3.6 Σερβοκινητήρες (Rc servos)

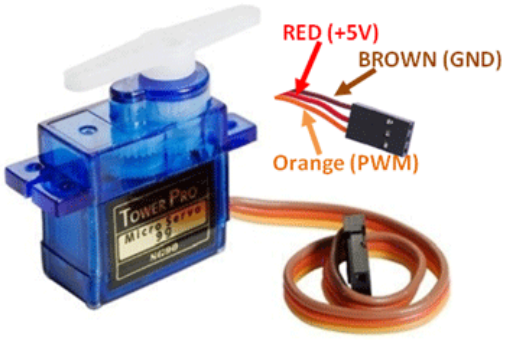
Οι μικροί αυτοί σερβοκινητήρες, είναι συνεχούς ρεύματος, διαθέτουν και σύστημα γραναζιών (συνήθως πλαστικά) για αύξηση της ροπής και μείωση των υψηλών στροφών που έχει ο κινητήρας, ενώ διαθέτουν επίσης και κύκλωμα ελέγχου ανάδρασης (feedback control loop) για να διατηρούν τη θέση τους. Χρησιμοποιούνται για να κινήσουμε μια άρθρωση, ένα τροχό ή οποιοδήποτε άλλο αντικείμενο επιθυμούμε. Για το μέγεθός τους παράγουν μια σχετικά μεγάλη ροπή και μπορούν να περιστραφούν με ταχύτητες της από 60 έως 160 rpm, και είναι κατάλληλοι για χρήση σε ρομποτικά συστήματα μικρής κλίμακας.

Εικόνα 18: Εσωτερική απεικόνιση σερβοκινητήρα

(Πηγή: Πτυχιακή εργασία Κριτσωτάκη Νικόλαου ΤΕΙ Μηχανολογίας Κρήτη 2012)

Εσωτερικά διαθέτουν ένα **κύκλωμα ελέγχου** που διαβάζει σήμα παλμού PWM ενώ επίσης έχουν ένα **ποτενσιόμετρο** το οποίο συνδέεται με τον άξονα, και όταν περιστρέφεται μεταβάλλεται η αντίσταση του ποτενσιομέτρου. Το κόστος τους είναι αρκετά χαμηλό και η συνδεσμολογία τους απλή, αφού απαιτούνται τρία μόλις καλώδια. Το κόκκινο δέχεται τάση +5V, το καφέ συνδέεται στα 0V ενώ στο κίτρινο καλώδιο συνδέουμε το PWM σήμα από τον ελεγκτή μας ή στη περίπτωσή μας από τον PCA9685 driver.

Το συγκεκριμένο ρομποτικό χέρι, περιλαμβάνει 5 τεμάχια σερβοκινητήρων micro sevo SG90. Το SG90 έχει τα εξής χαρακτηριστικά:



Εικόνα 19: Σερβοκινητήρας micro servo SG90

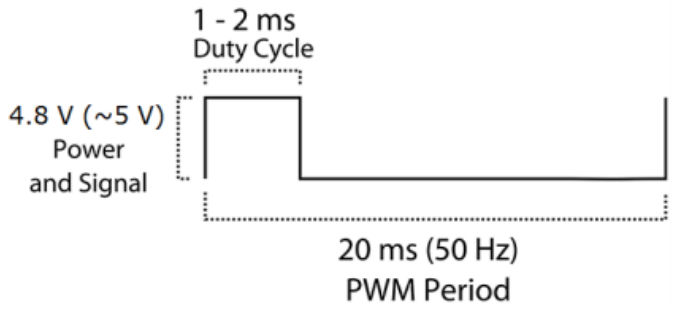
(Πηγή: https://components101.com/servo-motor-basics-pinout-datasheet)

Τεχνικά Χαρακτηριστικά

- Τάση τροφοδοσίας: 4.8 – 6.5V_{DC}
- Ροπή: 1.8kg/cm
- Περιστροφή: 0 – 180 μοίρες περίπου
- Βάρος : 9gr
- Ταχύτητα: 0.10 δευτερόλεπτα/60 μοίρες.
- Συχνότητα PWM: 50Hz (20msec)
- Πλάτος PWM παλμού: 500-2400μsec

Τρόπος λειτουργίας

Ο σερβοκινητήρας περιστρέφεται από 0 έως 180 μοίρες. Η συχνότητα PWM αντιστοιχεί σε 50Hz και το πλάτους του παλμού από τον μικροελεγκτή κυμαίνεται από 500-2400μsec.



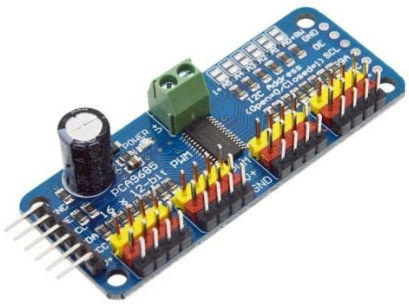
Εικόνα 20: Παλμός ελέγχου σερβοκινητήρα

(Πηγή: https://components101.com/servo-motor-basics-pinout-datasheet)

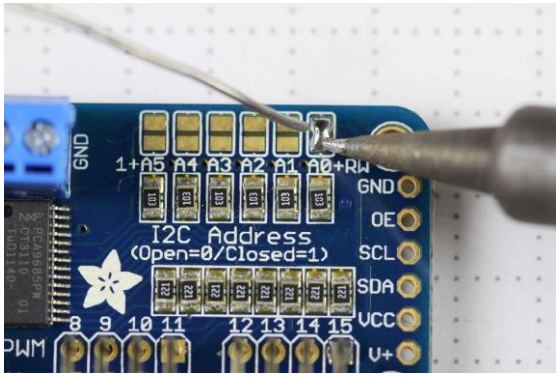
Συνήθως όταν το πλάτους του παλμού είναι περίπου 1msec ο σερβοκινητήρας πηγαίνει στις 0 μοίρες, όταν είναι 2msec πηγαίνει στις 180 μοίρες και για πλάτος παλμού 1.5msec ο σερβοκινητήρας πηγαίνει στις 90 μοίρες περίπου. Θα πρέπει να σημειωθεί ότι κάθε κινητήρας ακόμη και του ίδιου κατασκευαστή θα πρέπει να δοκιμαστεί και να καλιμπραριστεί για βρούμε τη σωστή αντιστοίχιση μεταξύ πλάτους παλμού ελέγχου και θέση σερβοκινητήρα (σε μοίρες).

3.7 PCA 9685 servodriver

Το PCA 9685 μπορεί να οδηγήσει έως και 16 σερβοκινητήρες με έξοδο PWM. Επικοινωνεί με το πρωτόκολλο I2C ενώ σε έναν master μπορούν να συνδεθούνε έως και 62 τέτοια modules. Η PWM έξοδος ρυθμίζεται για συχνότητες έως και 1.6KHz όπου η ανάλυση της κάθε εξόδου είναι 12bit για την ανωτέρω συχνότητα. Η τάση τροφοδοσίας είναι από 2.3 έως 5.5V ενώ τα υπόλοιπα Pins αντέχουν τάση έως 5.5V. Η αρχική διεύθυνση του PCA είναι η 0x40 ενώ μπορούμε να την αλλάξουμε από το πίσω μέρος όπου κολλάμε τις κατάλληλες γέφυρες (jumpers).



(Σύνδεσμος: grobotronics.com)



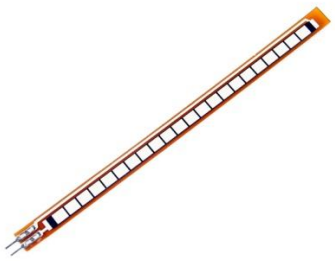
Εικόνα 21: Ο PCA 9685 servo driver

(Πηγή: https://osoyoo.com/2017/07/18/pca9685-16-channel-12-bit-pwm-servo-driver/)

Για τη χρήση σερβοκινητήρων μέσω του PCA 9685 διατίθεται **αντίστοιχη βιβλιοθήκη *(Adafruit_PWMServoDriver)*** την οποία πρέπει πρώτα να εγκαταστήσουμε μέσω του library manager στο IDE του Arduino.

3.8 Αισθητήρες κάμψης (Flex sensors)

Πρόκειται για μεταβλητή αντίσταση της τάξης των 30 έως 100ΚΩ (συνήθως). Έχουν πληθώρα εφαρμογών όπως σε βιντεοπαιχνίδια, ιατρικό εξοπλισμό, περιφερειακά Η/Υ, εφαρμογές ρομποτικής, ακόμη και σε επιφάνειες για την ένδειξη μη καθετότητας ή φθοράς.



Εικόνα 22: Αισθητήρας κάμψης μήκους 4.5 ιντσών

(Σύνδεσμος: amazon.in)

Η θερμοκρασία λειτουργίας τους είναι από -40° έως 85° C, και η ισχύς του από 0,5 έως 1W. Κάμπτοντας την εξωτερική επιφάνεια, η αντίσταση μεταβάλλεται ανάλογα με την κλίση ως προς τις μηδέν μοίρες δηλαδή στις 0 μοίρες η αντίσταση είναι περίπου 30ΚΩ ενώ στις 90 μοίρες, η αντίσταση αγγίζει τα 100ΚΩ. Η κλίμακα είναι γραμμική που σημαίνει ότι στις 45 μοίρες η αντίσταση θα είναι περίπου 65ΚΩ. Διατίθεται σε 2 μεγέθη, των 2.2 ιντσών και των 4.5 ιντσών μήκους. Για να μπορέσει ο χρήστης να επεξεργαστεί τις τιμές αυτές θα πρέπει να δημιουργήσουμε έναν διαιρέτη τάσης πρώτα για να μπορέσει ο μικροελεγκτής να διαβάσει τιμές τάσης (αναλογικές από 0 -5 V) και όχι τιμές αντίστασης.

3.9 HC05 Bluetooth module

Η ασύρματη επικοινωνία έχει εδραιωθεί σημαντικά τα τελευταία χρόνια και το Bluetooth είναι μια αξιόπιστη ασύρματη μέθοδος μεταφοράς μικρών σε όγκο πακέτων σε αποστάσεις έως **80 μέτρα** (ανάλογα με τη διαμόρφωση του χώρου). Χρησιμοποιείται συχνά σε περιφερειακά Η/Υ, παιχνίδια, ακουστικά κ.ο.κ.

Εικόνα 23: HC05 bluetooth master/slave module

(Πηγή: http://collegeroadonline.com/product/hc-05-bluetooth-module/)

Η συχνότητα μετάδοσης /λήψης είναι στα **2.45GHz**. Το HC05 ακολουθεί τα πρότυπα του **IEEE 802.15** που αφορά τις ασύρματες επικοινωνίες ενώ η επικοινωνία του με ελεγκτές ή άλλες συσκευές γίνεται μέσω σειριακού πρωτοκόλλου (**UART**) με baud rate έως 38.400 bits/sec . Το συγκεκριμένο module μπορεί να λειτουργήσει και ως master και ως slave.

3.10 Pi camera v2

Εικόνα 24: Picamera v2

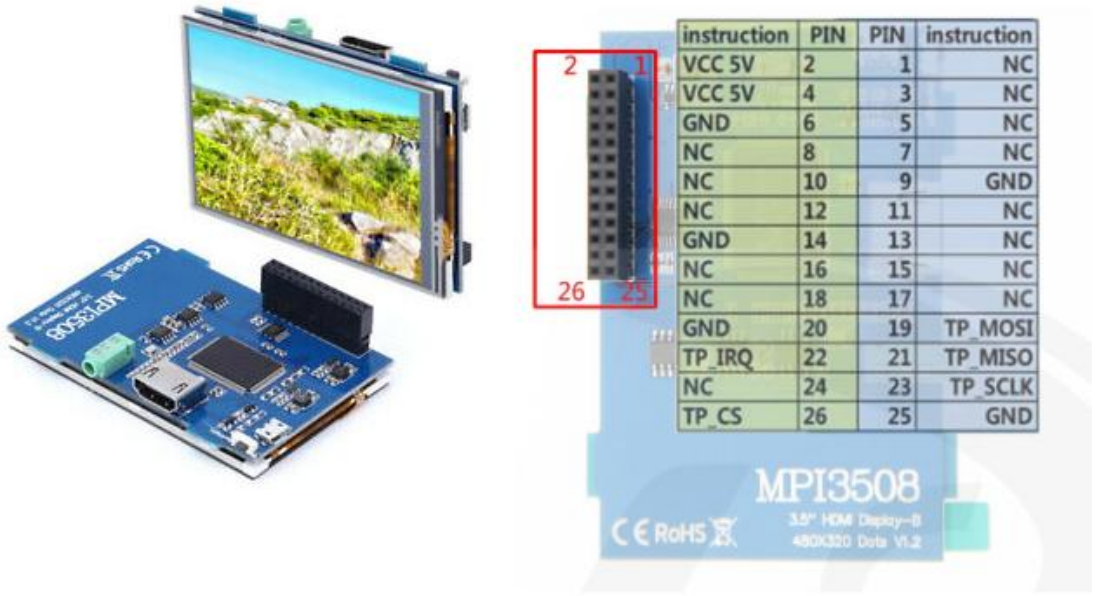
(Πηγή: https://www.antratek.com/raspberry-pi-8mp-camera-module-v2)

Η pi camera v2 διαθέτει αισθητήρα Sony IMX219 των 8 megapixels. Υποστηρίζει video σε φορματ 1080 p30, 720 p60 και VGA. Η pi camera βιβλιοθήκη διαθέτει πληθώρα λειτουργιών όπως η διαμόρφωση του frame rate, η ανάλυση, το φορματ των εικόνων που μπορεί να ‘’τραβήξει’’ η κάμερα καθώς και άλλες λειτουργίες.

Η επικοινωνία με το raspberry γίνεται μέσω της θύρας CSI (Camera Serial Interface πρωτόκολλο). Η ταχύτητα αποστολής των δεδομένων (εικόνων) στον επεξεργαστή του raspberry είναι της τάξης των Gbit/s.

3.11 LCD display MPI3508

Οθόνη 3.5 ιντσών που επικοινωνεί με raspberry pi μέσω SPI πρωτοκόλλου ενώ η απεικόνιση video γίνεται μέσω HDMI καλωδίου. Η ανάλυση είναι 480 x320 pixels (resistive touch screen). Αρχικά όμως πρέπει να γίνει εγκατάσταση του driver στην SD κάρτα μέσα από μια διαδικασία που περιγράφεται στο εγχειρίδιο χρήσης της οθόνης.



instruction	PIN	PIN	instruction
VCC 5V	2	1	NC
VCC 5V	4	3	NC
GND	6	5	NC
NC	8	7	NC
NC	10	9	GND
NC	12	11	NC
GND	14	13	NC
NC	16	15	NC
NC	18	17	NC
GND	20	19	TP_MOSI
TP_IRQ	22	21	TP_MISO
NC	24	23	TP_SCLK
TP_CS	26	25	GND

Εικόνα 25: MPI3508 Lcd οθόνη και τα σήματα σύνδεσής του

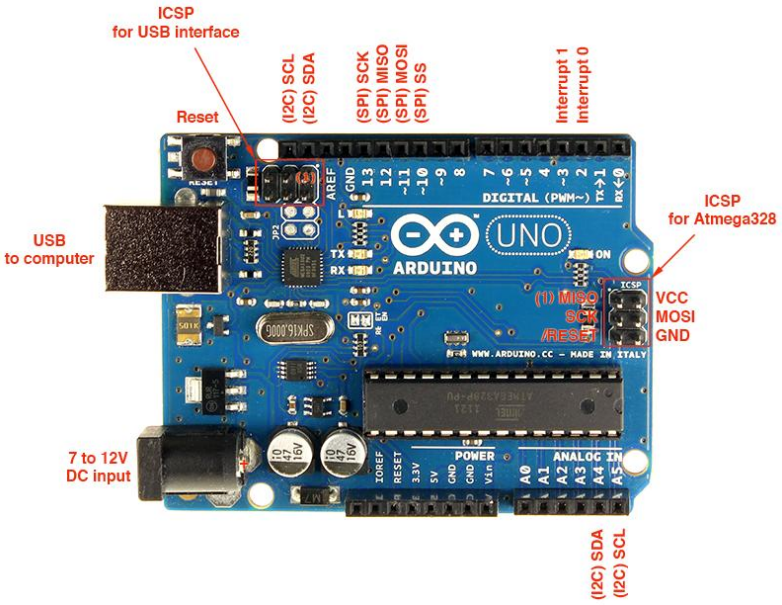
Σύνδεσμος: ebay.com (αριστερά)

Σύνδεσμος: http://www.lcdwiki.com/3.5inch_HDMI_Display-B (δεξιά)

Από τον κονέκτορα των 26 Pins χρησιμοποιηθήκανε τα 11 που χρειάζονται καθώς η απευθείας σύνδεση με το raspberry pi θα ‘’δεσμεύσει’’ αρκετά σήματα που είναι απαραίτητα για την διεκπεραίωση της εργασίας.

3.12 Arduino Uno –Nano

Ο Arduino είναι μία πλακέτα “ανοικτού κώδικα” με την οποία μπορεί κάποιος να φτιάξει εφαρμογές ρομποτικής και μικρά αυτόνομα συστήματα αυτοματισμού. Αποτελείται από έναν μικροεπεξεργαστή Atmega της Atmel και διαθέτει ψηφιακές εισόδους/εξόδους καθώς και αναλογικές για σύνδεση με αισθητήρες θερμοκρασίας , υγρασίας, δύναμης, απόστασης, επιταχυνσιόμετρα, γυροσκόπια κ.α. Επίσης μπορούμε να ελέγξουμε DC μοτέρ , βηματικούς κινητήρες (stepper motors), σερβοκινητήρες, ρελέ κ.α με τη χρήση των κατάλληλων drivers που θα τα τροφοδοτούνε με το απαιτούμενο ρεύμα. Τα boards (shields) αυτά, συνδέονται στα pin-headers του Arduino. Τα shields μπορούν να παρέχουν έλεγχο στα μοτέρ, GPS, Ethernet, LCD οθόνες και πληθώρα άλλων λειτουργιών.



Εικόνα 26: Arduino Uno

(Πηγή: https://www.electroschematics.com/arduino-uno-pinout/)

Ο Arduino μπορεί να προγραμματιστεί από έναν Η/Υ μέσω της σειριακής θύρας που διαθέτει. Η σειριακή αυτή σύνδεση (Serial over Usb) χρησιμοποιείται για την αποστολή των προγραμμάτων από τον υπολογιστή προς τον Arduino. Ο προγραμματισμός του Arduino είναι μία παραλλαγή της γλώσσας C/C++ αλλά υποστηρίζει όλες τις βασικές εντολές, δομές, συναρτήσεις της συγκεκριμένης γλώσσας ακόμη και αντικειμενοστραφή προγραμματισμό (Object Oriented Programming – OOP).

Περιβάλλον προγραμματισμού – Arduino IDE

Το περιβάλλον προγραμματισμού είναι το Arduino IDE (μπορούμε όμως να προγραμματίσουμε Arduino πλέον και σε Microsoft Visual Studio), το οποίο μεταγλωττίζει (compile) τον κώδικά μας σε γλώσσα μηχανής. Το Arduino IDE είναι δωρεάν και διατίθεται σε εκδόσεις για Windows, Mac και Linux. Στο περιβάλλον αυτό υπάρχουν έτοιμα παραδείγματα, library manager για χρήση έτοιμων βιβλιοθηκών καθώς και σειριακή απεικόνιση πληροφοριών ακόμη και γραφημάτων.

Εικόνα 27: Το περιβάλλον προγραμματισμού του Arduino

(Πηγή: https://en.wikipedia.org/wiki/Arduino_IDE)

Τεχνικά χαρακτηριστικά

Στον παρακάτω πίνακα παραθέτουμε τα τεχνικά χαρακτηριστικά των Arduino Uno/Nano.

Μικροελεγκτής	Atmega328P
Τάση λειτουργίας	5V
Τάση τροφοδοσίας (συνιστώμενη)	7-12V
Ψηφιακές είσοδοι/έξοδοι	14
PWM pins	6
Αναλογικές είσοδοι	6
Ταχύτητα ρολογιού	16MHz
Μνήμη FLASH	32KB
SRAM	2KB
EEPROM	1KB
Πρωτόκολλα επικοινωνιών	UART, SPI, I2C
Βάρος	25g

Πίνακας 1: Τεχνικά χαρακτηριστικά των Arduino Uno/Nano

Arduino Pins – Λειτουργίες

Το Arduino τροφοδοτείται με ρεύμα από τη USB θύρα 5V ή με παροχή 7-12V (Dc barrel jack). Υπάρχουν και παροχές 5 και 3.3V για τροφοδοσία εξωτερικών κυκλωμάτων και αισθητήρων. Διαθέτει επίσης 14 ψηφιακές εισόδους/εξόδους προγραμματιζόμενες μέσω του IDE.

Οι 14 ψηφιακές είσοδοι/έξοδοι έχουν αρίθμηση από το 0 έως το 13, ενώ οι έξι αναλογικές με το γράμμα Α ακολουθούμενο από ένα νούμερο από 0 μέχρι το 5. Να συμπληρώσουμε ότι κάποια από τα ανωτέρω Pins έχουν και δεύτερες ειδικές λειτουργίες. Για παράδειγμα τα I/O με αρίθμηση 3,5, 6, 9, 10 και 11 είναι και PWM θύρες (Pulse Width Modulation). Τα pins 10-13 μπορούνε να χρησιμοποιηθούνε και για σειριακή επικοινωνία SPI, τα pins A4 και 5 για σειριακή επικοινωνία I2C, και τα pins 0 και 1 για UART επικοινωνία. Θα πρέπει σε αυτό το σημείο να τονίσουμε ότι εάν ο Arduino επικοινωνεί σειριακά με οποιαδήποτε συσκευή μέσω USB καλωδίου και χρησιμοποιήσουμε και τα pins 0 και 1 (είτε ως ψηφιακές εισόδους/εξόδους είτε για UART επικοινωνία) τότε θα έχουμε σύγκρουση δεδομένων (conflict). Όλα τα pins αρχικοποιούνται μέσω της built in συνάρτησης PinMode().

Οι αναλογικές είσοδοι λειτουργούν από 0 έως 5V και έχουν ανάλυση 10 bit δηλαδή μια αναλογική μέτρηση μετατρέπεται σε μια ακέραια τιμή από 0 έως 1023 με ανάλυση 0.0049 V/μονάδα.

3.13 Arduino Nano

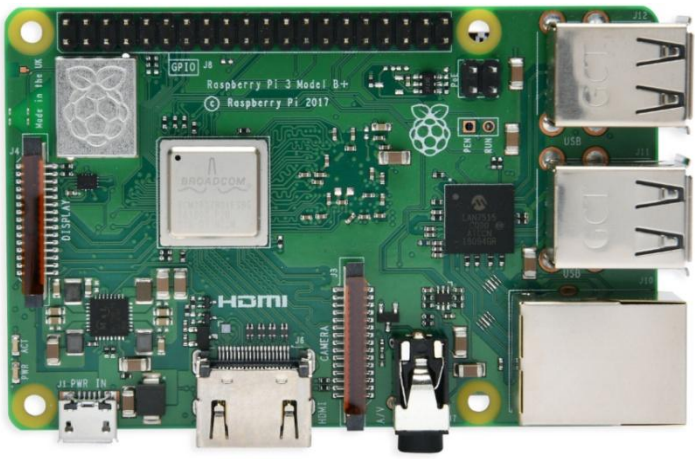
Για τον Arduino Nano δεν έχουμε να προσθέσουμε κάτι παραπάνω καθώς διαθέτει παρόμοια χαρακτηριστικά με τον Arduino Uno με τη μόνη διαφορά να είναι ότι ο Nano διαθέτει περισσότερες αναλογικές εισόδους ενώ ‘’βγαίνει’’ και σε μικρό μέγεθος για να μπορεί να τοποθετηθεί σε μικρά αυτόνομα συστήματα για συλλογή, επεξεργασία και μετάδοση πληροφοριών από αισθητήρες.

Εικόνα 28: Arduino Nano

(Πηγή: https://www.seeedstudio.com/Arduino-Nano-v3-p-1928.html)

3.14 Raspberry Pi 3B+

Το **Raspberry Pi** είναι ένας πλήρης υπολογιστής με ιδιαίτερα μέγεθος. Παρά τον μέγεθός του, το Raspberry Pi στην έκδοση 3B+ διαθέτει τετραπύρηνο επεξεργαστή στα 1200MHz, διπύρηνη κάρτα γραφικών, 1GB RAM, τέσσερις θύρες USB, έξοδο HDMI ανάλυσης έως 4K κ.α. Τροφοδοτείται με $5V_{DC}$ και διαθέτει 40 pins γενικής χρήσης για σύνδεση με άλλα ηλεκτρονικά και περιφερειακά εξαρτήματα.



Εικόνα 29: Raspberry Pi 3B+

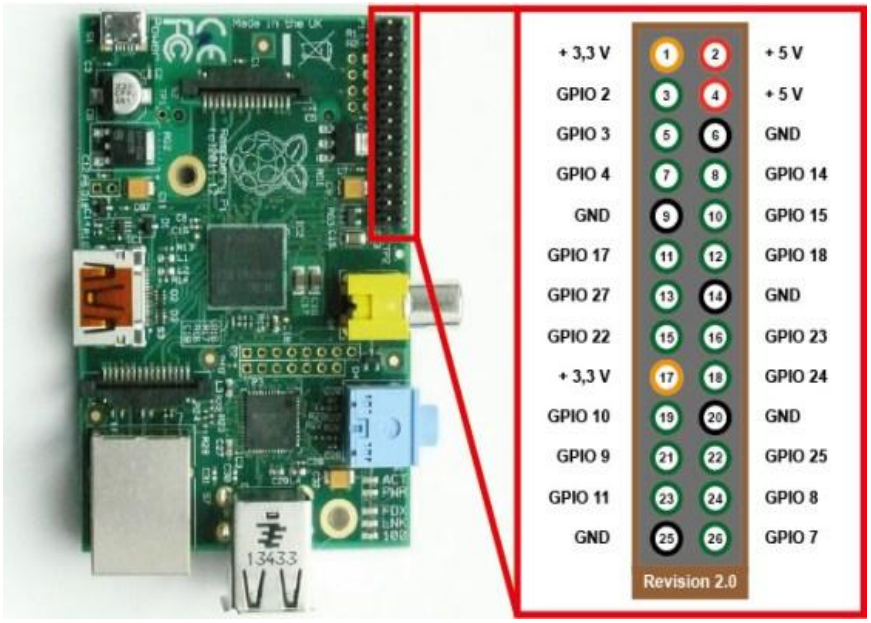
(Σύνδεσμος: amazon.com)

Το Raspberry Pi εμφανίστηκε το 2006, στο πανεπιστήμιο του Cambridge, όπου μια ομάδα εργαζομένων στο τμήμα Computer Laboratory το δημιούργησε με σκοπό την εισαγωγή των φοιτητών και απλών χρηστών στον κόσμο της πληροφορικής και των υπολογιστών, για αυτό το λόγο διατίθεται στην αγορά και σε προσιτή τιμή. Η ομάδα αυτή σχεδίασε αρκετά πρωτότυπα του Raspberry Pi αλλά το 2006 υπήρχαν περιορισμοί όπως το υψηλό κόστος και η χαμηλή επεξεργαστική ισχύς των επιμέρους ολοκληρωμένων κυκλωμάτων. Με την κυκλοφορία του πρώτου iPhone το 2007 και την δραματική αύξηση των πωλήσεων των έξυπνων τηλεφώνων (smartphones), το κόστος αυτό μειώθηκε σημαντικά. Το 2008 δημιουργήθηκε το φιλανθρωπικό ίδρυμα **Raspberry Pi Foundation** ενώ 3 χρόνια αργότερα εισήλθαν στην αγορά το πρώτο Raspberry Pi (Model A, Model A+ και Model B). Τα μοντέλα αυτά διέθεταν επεξεργαστή στα 700 MHz, 256MB RAM. Ακολούθησε η έκδοση Model B+ με 512MB RAM. Το 2015 κυκλοφόρησε το Raspberry Pi 2^{ης} γενιάς (Model B), με 1GB RAM και σημαντικά ταχύτερο επεξεργαστή, τον τετραπύρηνο Cortex-A7 (ARMv7). Το Νοέμβριο του 2015 κυκλοφόρησε το Raspberry Pi Zero, μία έκδοση με πολύ μικρό μέγεθος. Το Raspberry Pi 3^{ης} γενιάς κυκλοφόρησε το Φεβρουάριο του 2016 και είναι εφοδιασμένο με ταχύτερο επεξεργαστή ARM Cortex-A53 στα 1200MHz, 1GB RAM και κάρτα γραφικών Broadcom VideoCore IV χρονισμένη στα 250MHz. Από το 2019 κυκλοφορεί και η τελευταία γενιά, η 4^{η} για την ακρίβεια η οποία διαθέτει έως και 8GB DDR4 RAM, USB 3.0 θύρες, τετραπύρηνο επεξεργαστή 1.5GHz και μπορεί να συγκριθεί εύκολα ακόμη και με desktop Η/Υ.

Το **λειτουργικό του σύστημα ονομάζεται raspbian** και βασίζεται σε λειτουργικό σύστημα linux αλλά βελτιστοποιημένο αποκλειστικά για το Raspberry Pi. Το raspbian ‘’βρίσκεται’’ στην Micro SD. Όπως κάθε linux σύστημα έτσι και στο raspbian ο χρήστης θα πρέπει να είναι εξοικειωμένος με εντολές, (μέσω του command terminal) για την πλοήγηση, δημιουργία και επεξεργασία φακέλων, εγκατάσταση πακέτων και άλλων λειτουργιών.

Το Raspberry Pi χρησιμοποιείται ευρέως στον τομέα της **εκπαίδευσης, σε οικιακούς αυτοματισμούς** (και λόγω της ενσωματωμένης κάρτας δικτύου για απομακρυσμένο έλεγχο, σε αντίθεση με έναν μικροελεγκτή), **ρομποτικές εφαρμογές, ακόμη και σε βιομηχανικές εφαρμογές** (π.χ χρήση ενός Raspberry Pi για δημιουργία VPN δικτύου για επίβλεψη μιας παραγωγικής μονάδας κτλ).

Τα pins του Raspberry Pi (**GPIO**) είναι γενικού σκοπού τα περισσότερα, ενώ διατίθενται και **ειδικού σκοπού όπως pins για επικοινωνία I2C, SPI, UART καθώς και PWM σήματα**. Η αρίθμηση των pins αυτών γίνεται με 2 τρόπους. Ο πρώτος είναι με την φυσική αρίθμηση δηλαδή από αριστερά στα δεξιά και από πάνω προς τα κάτω. Ο δεύτερος και πιο συνηθισμένος είναι με την **BCM αρίθμηση** του ολοκληρωμένου και αποτυπώνεται στην παρακάτω εικόνα.



Εικόνα 30: Αρίθμηση και λειτουργίες των pins του Raspberry Pi 3B+

(Πηγή: https://www.marcelpost.com/wiki/index.php/GPIO_pin_on_the_RPi_to_trigger_PTT)

Τεχνικά χαρακτηριστικά Raspberry Pi 3 B+

- Τάση τροφοδοσίας $5V_{DC}$ / 2.5A
- Λειτουργία σε θερμοκρασία περιβάλλοντος 0-50°C
- Επεξεργαστή Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC 1.4GHz
- 1GB LPDDR2 SDRAM
- Ασύρματη επικοινωνία 2.4GHz και 5GHz IEEE 802.11.b/g/n/ac wireless LAN, καθώς και Bluetooth 4.2, BLE
- Gigabit Ethernet
- 40 pin GPIO για σύνδεση εισόδων/εξόδων
- HDMI θύρα
- 4 θύρες USB 2.0
- CSI θύρα για σύνδεση καμερών
- DSI display port for connecting a Raspberry Pi touch screen display
- Micro SD θύρα για το λειτουργικό σύστημα καθώς και για αποθήκευση

3.15 Πίνακας υλικών κατασκευής

Για το γάντι χρησιμοποιήθηκαν τα εξής υλικά:

Υλικό	**Ποσότητα(τεμάχια)**
Γάντι νιτριλίου	1
Powerbank $5V_{DC}$ 2Ah	1
Αισθητήρες κάμψης μήκους 4.5’’	5
HC05 bluetooth module	1
Ρυθμιστής τάσης από 5 σε 3.3V	1
Arduino Nano	1
Led 5mm (κόκκινο, πράσινο)	2
Καλώδια	1

Πίνακας 2: Υλικά κατασκευής γαντιού

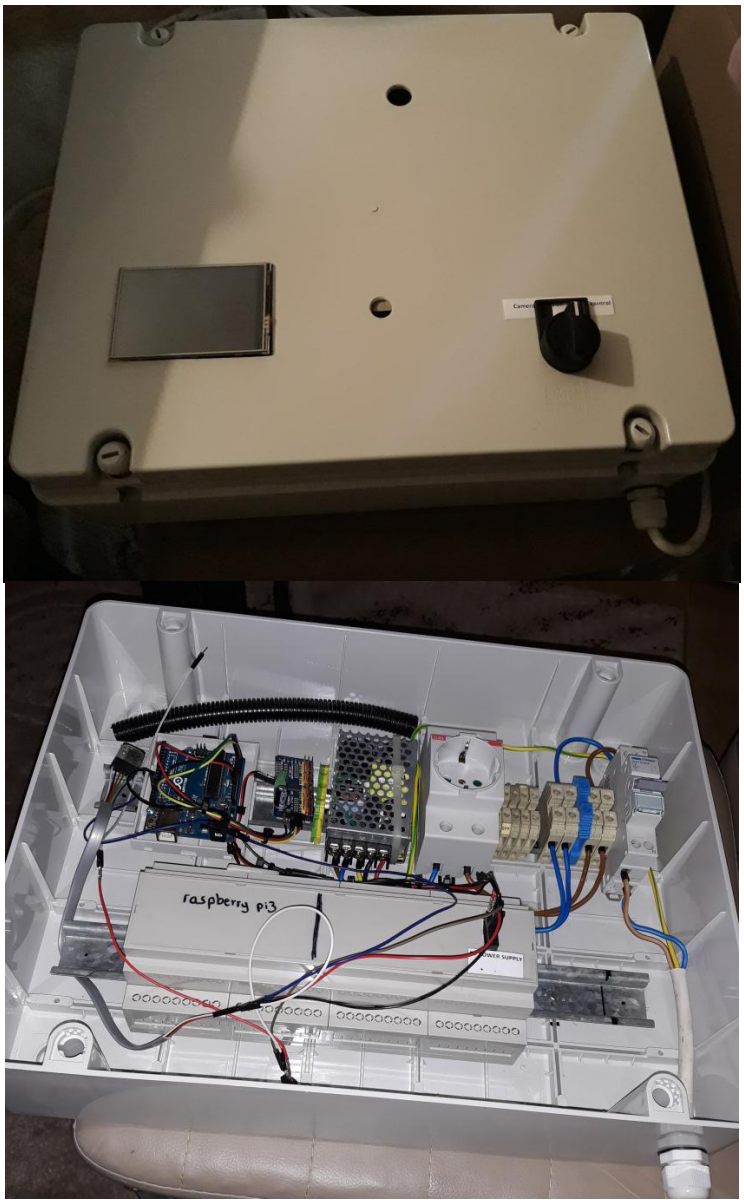
Εικόνα 31: Το απτικό γάντι με ενσωματωμένα τους αισθητήρες κάμψης, τον μικροελεγκτή και το Bluetooth module

Αντίστοιχα για τη βάση χρησιμοποιήθηκε ένα πλαστικό ηλεκτρολογικό κουτί με τα εξής υλικά:

Υλικό	**Ποσότητα(τεμάχια)**
Ηλεκτρολογικό κουτί GEWISS διαστάσεων 50x40x30	1
Μεταλλική βάση στήριξης ρομποτικού χεριού	1
Ρομποτικό χέρι με 5 σερβοκινητήρες	1
Επιλογικός διακόπτης 0-1-2	1
Μπουτόν N.O για επανεκκίνηση του Raspberry	1
Pi camera v2 με καλώδιο μήκους 1mt για σύνδεση με CSI θύρα.	1
LCD οθόνη MPI3508 3.5’’	1
HDMI καλώδιο	1
Ασφαλειοδιακόπτης διπολικός 10A	1
Κλέμμες	15
DIN ράγα	2
Τροφοδοτικό $5V_{DC}$ 5A	1
Τροφοδοτικό $5V_{DC}$ (για πρίζα) Raspberry Pi 2.5A	1
Πρίζα ράγας	1
Arduino Uno	1
Raspberry Pi 3 B+	1
HC05 bluetooth module	1
PCA 9685 servo driver	1

Ασύρματο Bluetooth πληκτρολόγιο για χειρισμό του Raspberry Pi	1
Ρυθμιστής τάσης από 5 σε 3.3V	1
Ρελέ με πηνίο τάσης 3-12V_{DC}	1
NPN τρανζίστορ 2N3904	1
Δίοδος	1
Αντίσταση 1ΚΩ	1
Καλώδια	1

Πίνακας 3: Πίνακας υλικών κατασκευής



Εικόνα 32: Η πορεία της κατασκευής της βάσης του ρομποτικού χεριού

Κεφάλαιο 4

4.1 Εισαγωγή – Ερευνητική μέθοδος

Στο κεφάλαιο αυτό θα παρουσιάσουμε την μεθοδολογία και τον τρόπο που εργαστήκαμε για την υλοποίηση της συγκεκριμένης διπλωματικής. Θα αναφερθούμε στο ηλεκτρολογικό και ηλεκτρονικό κομμάτι (αισθητήρες, ρελέ, ελεγκτές κτλ.) καθώς και στον **αλγόριθμο υλοποίησης ο οποίος όμως περιλαμβάνει κομμάτια και δεν παρουσιάζεται ολόκληρος σε αυτό το σημείο.**

Επειδή η διπλωματική αποτελείται από 2 διαφορετικούς τρόπους λειτουργίας θα αναλυθούν ξεχωριστά.

Οι δυο αυτοί τρόποι λειτουργίας (modes ελέγχου) είναι οι εξής:

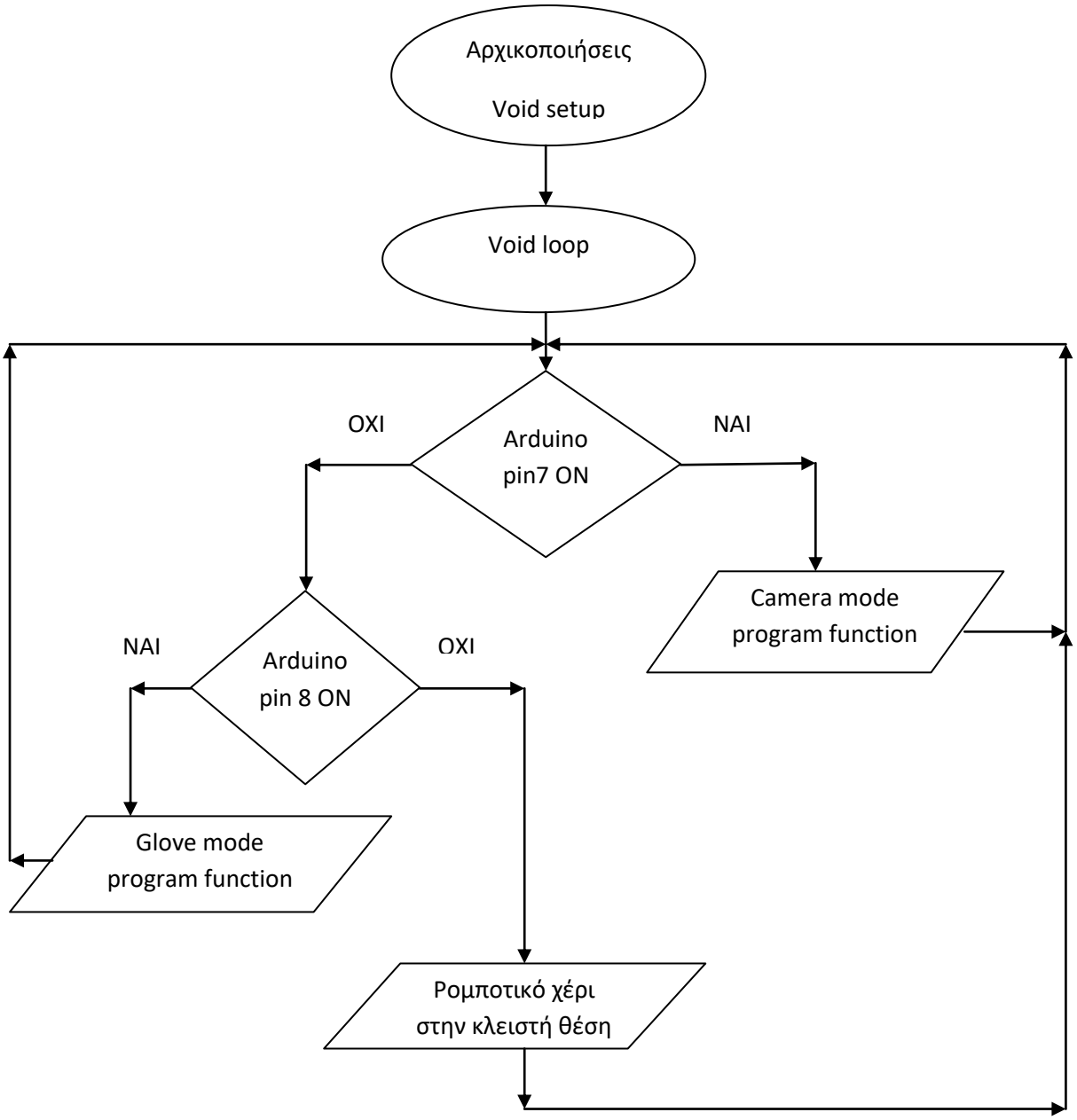
1) Έλεγχος ρομποτικού χεριού με γάντι (νιτριλίου που κυκλοφορούν στο εμπόριο) μέσω ασύρματης επικοινωνίας Bluetooth.
2) Αναγνώριση ανθρώπινου χεριού και αντίστοιχη κίνηση του ρομποτικού χεριού μέσω κάμερας και Raspberry Pi.

Η επιλογή του κάθε mode λειτουργίας γίνεται από τον επιλογικό διακόπτη 0-1-2 όπου στη θέση 2 (δεξιά) εκτελεί τον έλεγχο του προγράμματος για τον έλεγχο του ρομποτικού χεριού με το γάντι (πρόγραμμα 2). Στη θέση 1 (αριστερά) εκτελεί το πρόγραμμα για την αναγνώριση μέσω κάμερας (πρόγραμμα 1). Επίσης, έχει τοποθετηθεί power on μπουτόν που συνδέεται στην ψηφιακή είσοδο 5 (κατά την BCM αρίθμηση) όπου όταν γίνει τερματισμός λειτουργίας του Raspberry και πατηθεί, θα εκκινήσει εκ νέου το Raspberry.

Παρακάτω παρατίθεται το ηλεκτρολογικό σχέδιο καθώς και φωτογραφίες της κατασκευής.

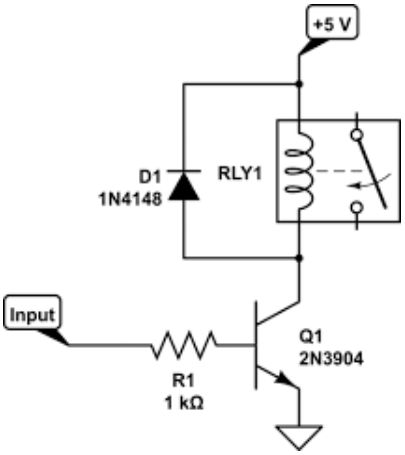
Εικόνα 33: Η βάση του ρομποτικού χεριού

Αξίζει να σημειώσουμε ότι το Raspberry και το Arduino επικοινωνούνε σειριακά μέσω UART (Universal Asynchronous Receiver – Transmitter) πρωτοκόλλου επικοινωνίας. Όταν χρησιμοποιείται η θύρα USB για την σειριακή επικοινωνία τότε δεν μπορούν να χρησιμοποιηθούνε τα pins 0 και 1 του Arduino Uno. Στην περίπτωσή μας, το HC-05 (slave) που βρίσκεται εντός της βάσης, λαμβάνει συνεχώς πακέτα από το HC-05 (master) που βρίσκεται στο γάντι, για να τα στείλει στον Arduino για επεξεργασία. Τα πακέτα αυτά, στέλνονται και αυτά σειριακά μέσω UART στον Arduino Uno. Επιπλέον, όταν ο χρήστης επιλέξει το πρόγραμμα 2 (αναγνώριση με κάμερα) το Raspberry στέλνει πακέτα σειριακά από τη USB θύρα. Όπως προκύπτει από τα παραπάνω, δεν μπορούμε να έχουμε λήψη ή αποστολή δεδομένων από την ίδια UART θύρα ταυτόχρονα για τον Arduino Uno. Συνεπώς θα πρέπει να αποκόπτουμε τη μεταφορά πακέτων από το HC-05 στο Arduino όσο ο επιλογικός διακόπτης βρίσκεται στη θέση 1 (έλεγχος μέσω κάμερας).



Γράφημα 4: Λογικό διάγραμμα αλγορίθμων εκτέλεσης

Το Raspberry τρέχει το python script όπου εάν ο χρήστης έχει επιλέξει το πρόγραμμα 1 ενεργοποιεί την έξοδο 6 (BCM αρίθμηση) που ενεργοποιεί με τη σειρά του το ρελέ με τάση 3-12V_{DC}. Οι επαφές N.C του ρελέ συνδέουν τα Receive και Transmit σήματα από το HC-05 στο Arduino, οπότε όταν διεγερθεί το ρελέ αποκόπτονται τα σήματα αυτά που καταλήγουνε στα pins 0 και 1 του Arduino, για να στέλνονται μόνο από τη usb θύρα του Raspberry. Επειδή όμως η έξοδος του Raspberry δεν μπορεί να οπλίσει το ρελέ απ' ευθείας, έχει τοποθετηθεί ένα NPN τρανζίστορ και μια δίοδος γιατί όταν ενεργοποιείται ένα πηνίο δημιουργείται αντι-Η.Ε.Δ (αντίθετης φοράς ηλεκτρεγερτική δύναμη ή back emf) που μπορεί ακόμα και να κάψει την έξοδο του Raspberry. Τα ανωτέρω συνοψίζονται στο παρακάτω κύκλωμα.



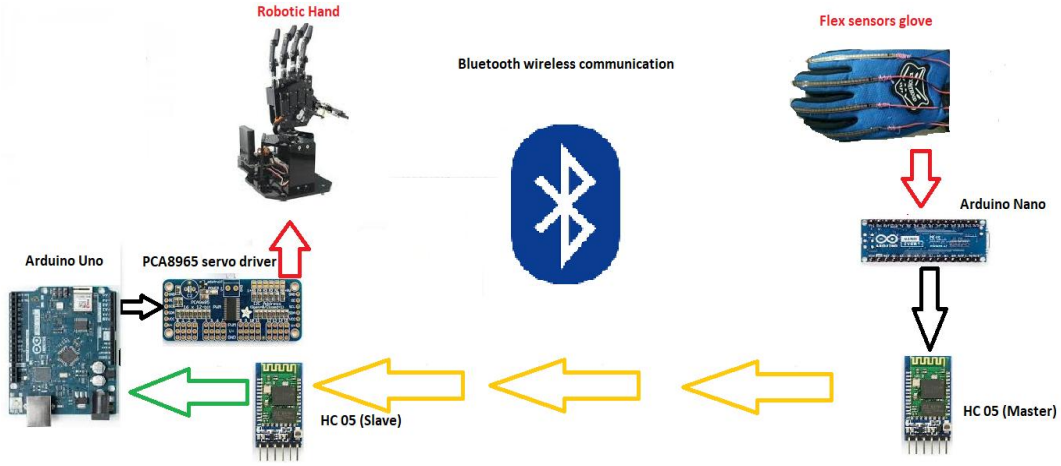
Εικόνα 34: Κύκλωμα ελέγχου ρελέ για τη σειριακή επικοινωνία

(Πηγή: https://www.raspberrypi.org/forums/viewtopic.php?t=180554)

4.2 Έλεγχος ρομποτικού χεριού μέσω απτικού γαντιού (1^{ο} Mode ελέγχου)

4.2.1 Λειτουργία

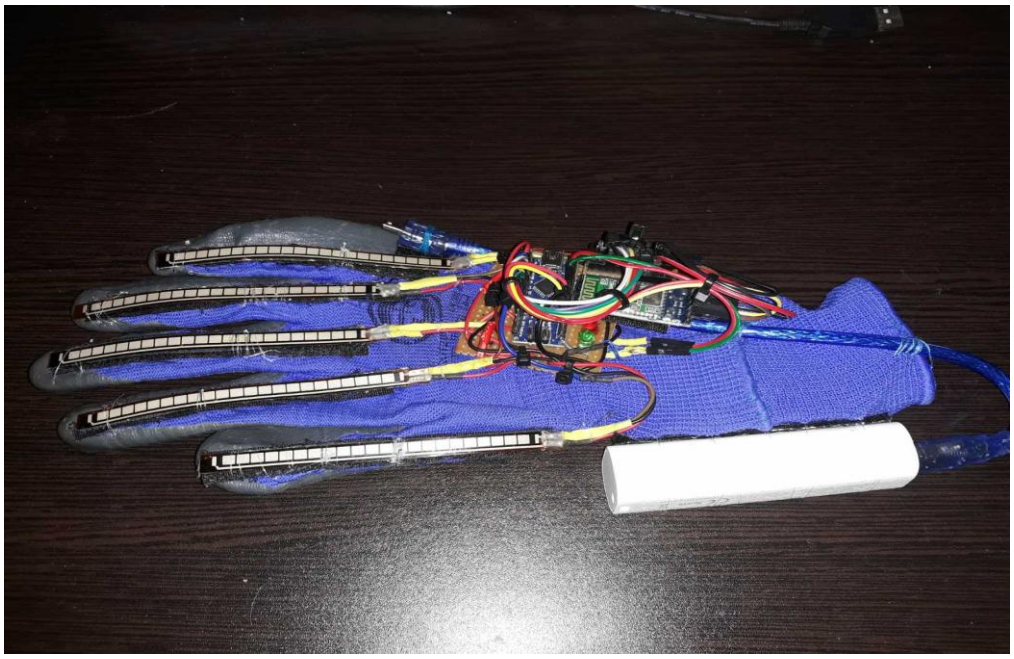
Το πρόβλημα μπορεί ουσιαστικά να χωριστεί σε 2 τμήματα. Το ένα είναι του γαντιού όπου διαβάζουμε τις τιμές των αισθητήρων κάμψης, τα επεξεργαζόμαστε, τα φιλτράρουμε και τα στέλνουμε σειριακά στο HC05 module (Master) το οποίο συνεχώς στέλνει τις αναλογικές τιμές στην βάση. Το δεύτερο τμήμα αποτελείται από το ρομποτικό χέρι που ελέγχεται από έναν δεύτερο Arduino (Uno), ο οποίος διαβάζει σειριακά τα πακέτα των αναλογικών τιμών που λαμβάνει από ένα δεύτερο HC05 module (slave) και αντίστοιχα στέλνει τα κατάλληλα σήματα ελέγχου κίνησης στους σερβοκινητήρες μέσω του PCA9685 servo module το οποίο μπορεί να ελέγξει έως και 16 σερβοκινητήρες.



Εικόνα 35: Απεικόνιση λειτουργίας

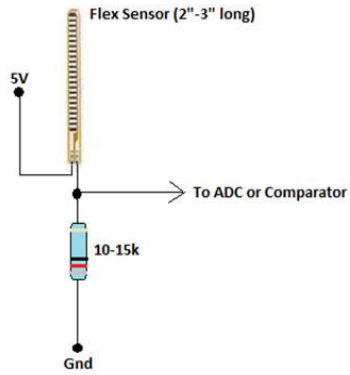
4.2.2 Κατασκευή

Αρχικά έγινε η κατασκευή του γαντιού. Οι αισθητήρες κάμψης έχουν τοποθετηθεί με Velcro και έχουν ραφτεί πάνω στο γάντι λόγω ευκολίας. Επειδή όμως ο ελεγκτής μας, ο οποίος είναι ο Arduino Nano (επιλέχθηκε λόγω μεγέθους) δεν διαβάζει απ' ευθείας Ω (αντίσταση) πρέπει να γίνει κύκλωμα μετατροπής σε αναλογικά σήματα 0-5V με τη χρήση αντιστάσεων (κύκλωμα διαιρέτη τάσης).



Εικόνα 36: Τελικό στάδιο κατασκευής του γαντιού ελέγχου

Παρακάτω βλέπουμε το κύκλωμα του διαιρέτη τάσης για κάθε δάχτυλο που θα καταλήγει στις αναλογικές εισόδους A1-A5 του Arduino Nano.



Εικόνα 37: Διαιρέτης τάσης 0-5V για τον αισθητήρα κάμψης.

(Πηγή: https://www.engineersgarage.com/contributions/interfacing-flex-sensor-with-arduino/)

Αφού διαβάσουμε όλες τις αναλογικές τιμές για κάθε δάχτυλο βλέπουμε ότι η μέγιστη τιμή δεν ξεπερνάει το 550 (δηλ. περίπου 2.7V). Για να μπορέσουμε να μειώσουμε το χρόνο αποστολής των πακέτων αυτών στο HC-05 που θα στέλνει συνεχώς αυτές τις τιμές, διαιρέσαμε με μια σταθερά (2.35 για την ακρίβεια) ώστε να στέλνουμε πακέτα των byte (με μέγιστη τιμή 255) και όχι των integer (μέγιστη τιμή 65.535).

4.2.4 Λειτουργία HC05 bluetooth module

Για την ασύρματη μετάδοση των αναλογικών τιμών από το γάντι προς το Arduino που ελέγχει το ρομποτικό χέρι, μπορεί να γίνει είτε μέσω Bluetooth είτε μέσω wifi (π.χ esp 8266) είτε μέσω των NRF24L01 (ράδιο κύματα συχνότητας 2.4-2.5GHz). Επιλέξαμε το πρώτο καθώς καταναλώνει σημαντικά λιγότερη ενέργεια και σε ένα αυτόνομο σύστημα με μπαταρίες η αυτονομία είναι πολύ σημαντικός παράγοντας.

Πρέπει να σημειωθεί ότι τα σήματα Receive / Transmit (Rx/ Tx) είναι της τάξης των 3.3V. Τα αντίστοιχα σήματα στον Arduino όμως έχουν τάση 5V. Όταν συνδεθεί το Tx του HC-05 με το Rx τουArduino τότε δεν υπάρχει πρόβλημα καθώς το Arduino εκλαμβάνει ως ‘’1’’ τη στάθμη των 3.3V. Στην σύνδεση του Rx του HC-05 με το Tx του Arduino όμως χρειάζεται να ρίξουμε την τάση των 5V σε 3.3V για να μην το κάψουμε. Αυτό μπορεί να γίνει με έναν απλό

διαιρέτη τάσης ή με έναν ρυθμιστή τάσης από 5 σε 3.3V όπως ο LD1117, όπως και τελικά επιλέξαμε. Ο ρυθμιστής τάσης προτιμήθηκε καθώς δεν σπαταλάει ενέργεια ως θερμότητα από τις αντιστάσεις, πρόβλημα όπου είναι πιο σημαντικό στο HC-05 που τοποθετήθηκε στο γάντι και λειτουργεί μέσω Lipo μπαταρίας όπου δεν θέλουμε άσκοπη κατανάλωση ρεύματος (έστω και μικρή).

Για να μπορέσουμε να έχουμε αμφίδρομη επικοινωνία μεταξύ των δυο HC-05 modules, πρέπει αρχικά να ρυθμιστούνε κάποιες παράμετροι όπως το baud rate, εάν θα είναι master ή slave, όνομα του module κ.α.

Εικόνα 38: HC 05 bluetooth module pins

(Πηγή: https://beatyourbit.com/?s=hc05)

EN (enable)	Όταν είναι σε λογικό ‘0’ το module είναι σε λειτουργία αποστολής/λήψης δεδομένων. Όταν είναι σε λογικό ‘1’ το module είναι σε λειτουργία AT (αρχικοποίησης).
VCC	Τροφοδοσία HC05 module με 5V
GND	0V
TXD	Pin αποστολής σειριακών δεδομένων, τάσης 3.3V
RXD	Pin λήψης σειριακών δεδομένων, τάσης 3.3V
STATE	Έξοδος (ως feedback)

Πίνακας 4: Επεξήγηση Pins

4.2.5 Ρύθμιση παραμέτρων HC05

Για να μπορέσουμε να ρυθμίσουμε τις παραμέτρους στο HC-05 bluetooth module θα πρέπει πρώτα να κατεβάσουμε το εξής πρόγραμμα και έπειτα να στέλνουμε μερικές συγκεκριμένες AT εντολές μέσω ενός Arduino σειριακά. Συνδέουμε τα Pins 9,10 στα Tx, Rx του HC 05.

```
#include <SoftwareSerial.h>

Const int txPin = 9;
Const int rxPin = 10;

SoftwareSerial  BTSerial(rxPin,  txPin);  // RX, TX

void setup() {
  Serial.begin(9600);
  Serial.println("Enter AT commands:");
  BTSerial.begin(38400);
}
void loop() {
   if (BTSerial.available())
       Serial.write(BTSerial.read());
   if (Serial.available())
       BTSerial.write(Serial.read());
}
```

Ουσιαστικά ‘’δημιουργούμε’’ μια εικονική σειριακή θύρα μέσω της βιβλιοθήκης Software Serial στα Pins 9 και 10, ώστε να στέλνουμε από αυτήν τις AT εντολές προς το HC 05 και μέσω της hardware σειριακής (USB) που διαθέτει ο Arduino να μπορούμε να βλέπουμε εάν έχουν ληφθεί αυτές οι εντολές.

Έπειτα, στο serial monitor, ρυθμίζουμε το baud rate στα 9600 bps και επιλέγουμε CR+NL ως χαρακτήρας τέλους και γράφουμε ‘’AT + Εντολή’’ όπου έχουμε τις εξής:

HC05 Master (γάντι)

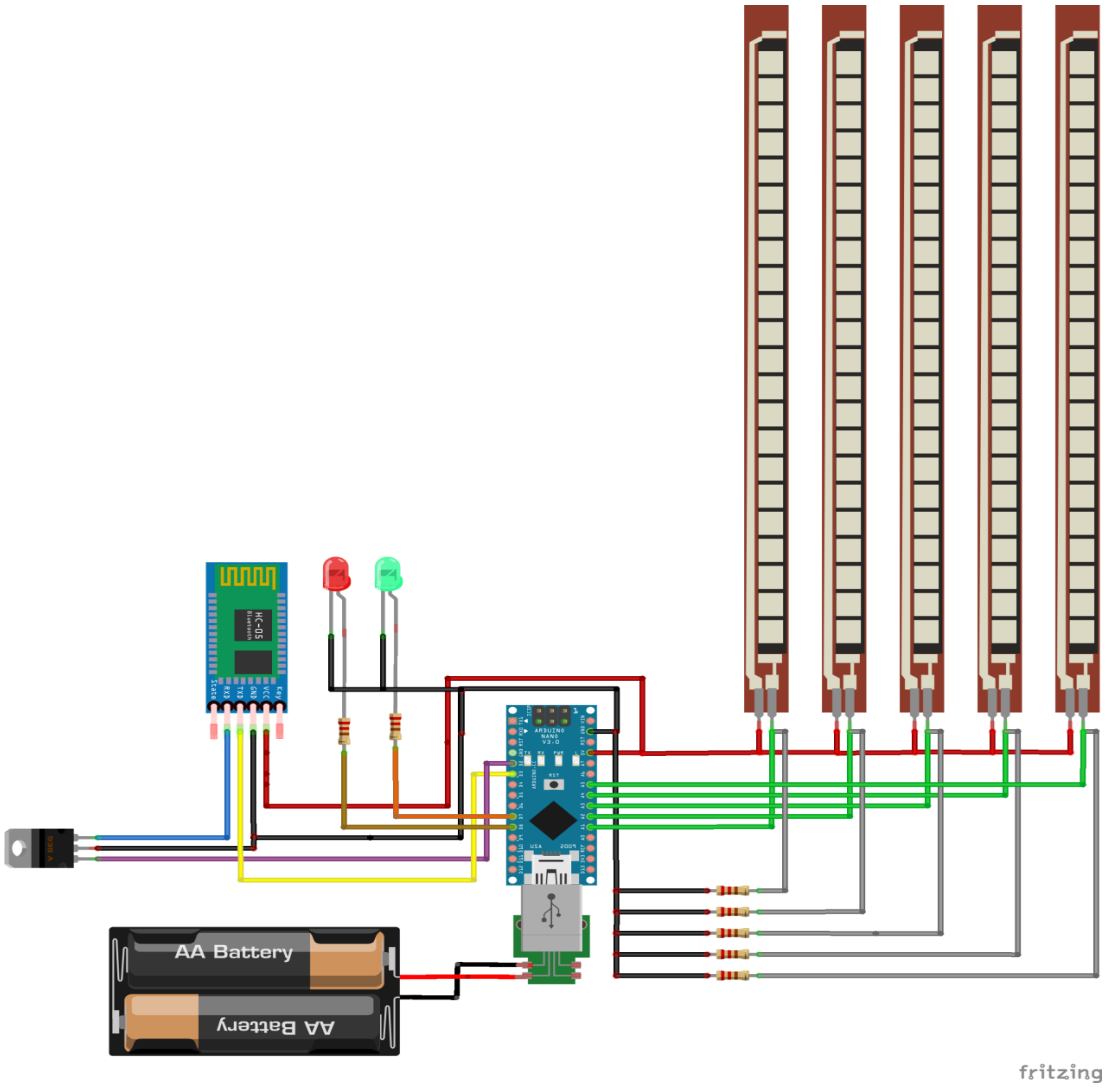
- AT+NAME=HC_05_MASTER (αλλάζουμε το όνομα)
- AT+NAME? Διαβάζουμε όνομα
- AT+ROLE =1 (Master)
- AT+ROLE? Εάν είναι 0 σημαίνει ότι είναι slave. Εάν είναι 1 σημαίνει ότι είναι master
- AT+UART=38400,0,0 baud rate, parity bits
- AT+ADDR? Διαβάζουμε τη διεύθυνση του master

- AT+BIND = Διεύθυνση του HC05 slave με το οποίο θα συνδεθεί

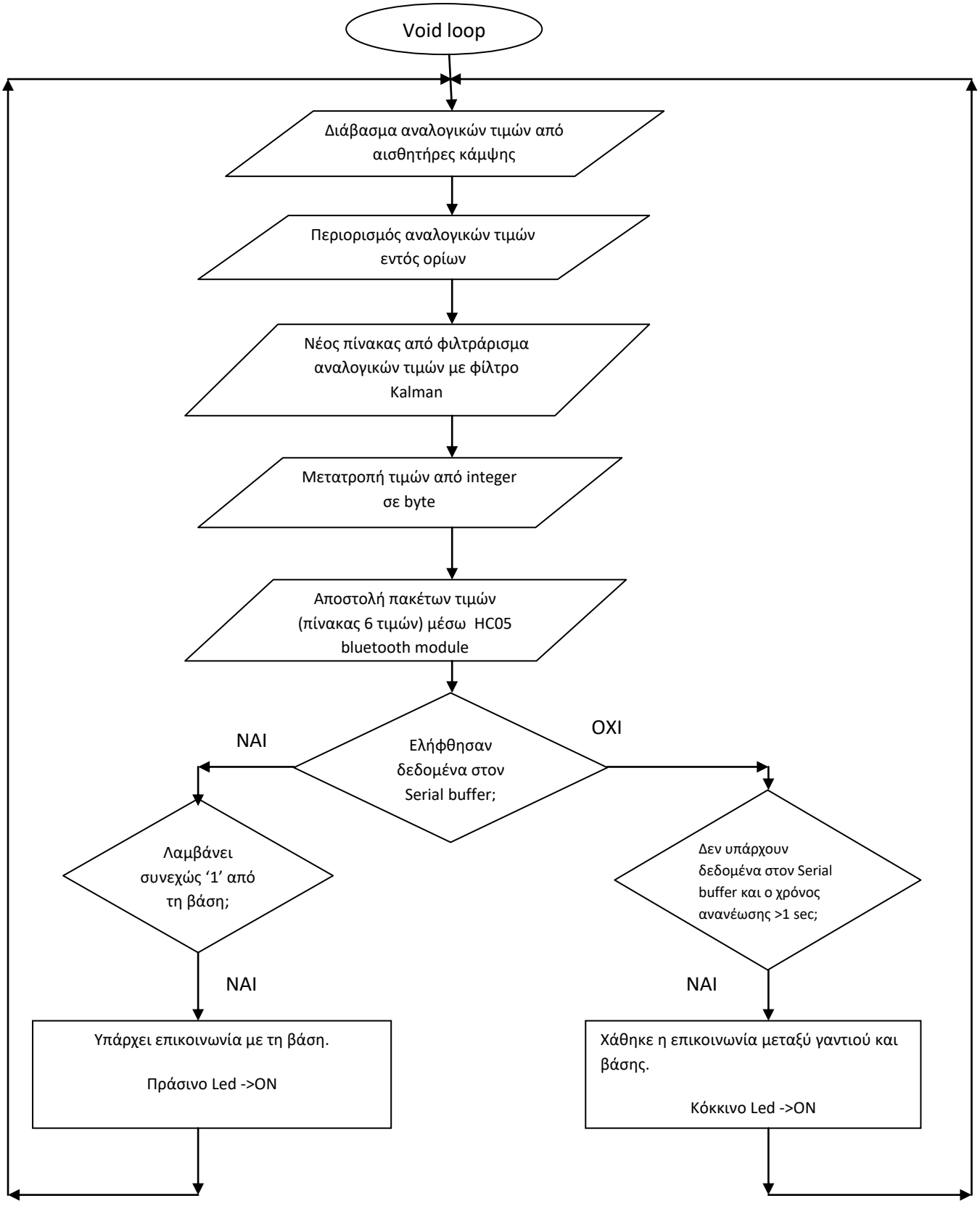
HC05 Slave (ρομποτικό χέρι)

- AT+RMAAD -> Διαγράφει όποια σύνδεση παλαιότερη υπήρχε με master
- AT+NAME=HC_05_SLAVE (αλλάζουμε το όνομα)
- AT+UART=38400,0,0 baud rate, parity bits
- AT+ROLE =0 (Slave)

Αφού γίνει και η ρύθμιση των HC 05 Bluetooth modules, παραθέτουμε το διάγραμμα ροής, το ηλεκτρικό κύκλωμα του γαντιού καθώς και τον κώδικα για το απτικό γάντι.



Εικόνα 39: Ηλεκτρολογικό σχέδιο απτικού γαντιού



Γράφημα 5: Λογικό διάγραμμα γαντιού

Στη συνέχεια πρέπει να διαβάσουμε τα όρια των αναλογικών τιμών για την έκταση και σύμπτυξη των δαχτύλων. Έπειτα από μετρήσεις στη κλειστή θέση (γροθιά) και στην ανοιχτή (παλάμη) καταλήξαμε στις ακόλουθες μετρήσεις. Οι μέγιστες-ελάχιστες τιμές της εντολής Analog Read είναι από 0 έως 1023 (5V), ανάλυσης 10 bit.

Δάχτυλο	**Κάτω όριο**	**Άνω όριο**
Αντίχειρας (Thumb)	**305**	**450**
Δείκτης (Index)	**110**	**185**
Μεσαίος (Middle)	**238**	**445**
Παράμεσος (Ring)	**235**	**400**
Μικρός (Pinky)	**260**	**453**

Πίνακας 5: Τιμές τάσης αισθητήρων κάμψης για τα δάχτυλα

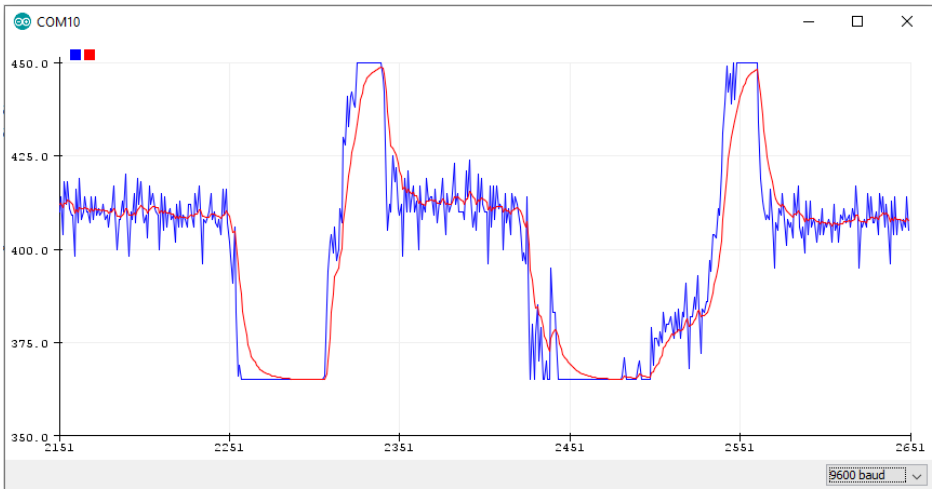
Επιπλέον χρειάστηκε να περιορίσουμε τις αναλογικές τιμές με την εντολή constrain εντός κάποιων ορίων.

Αρχικά στέλναμε στον Arduino Uno της βάσης τις τιμές αυτές αλλά παρατηρήθηκε ότι οι αισθητήρες κάμψης περιέχουν και θόρυβο αλλά και είναι ιδιαίτερα ευαίσθητοι στις μικρές μετατοπίσεις με αποτέλεσμα τα δάχτυλα του ρομποτικού χεριού να εμφανίζουν μια διαρκή ‘’νευρική’’ κίνηση (jitter).

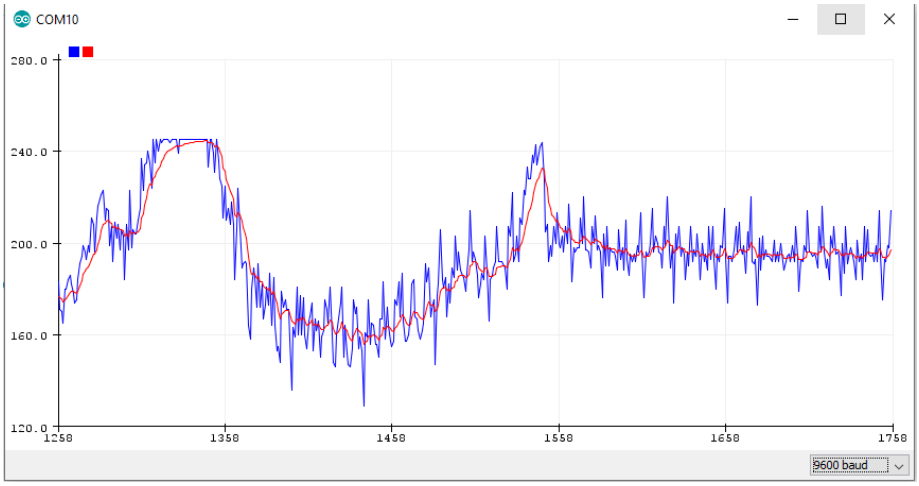
Για αυτό το λόγο επιλέξαμε την εφαρμογή του Kalman φίλτρου για καλύτερο έλεγχο των σερβοκινητήρων.

Παρακάτω παρατίθενται οι αναλογικές τιμές και των 5 δαχτύλων ως διάγραμμα στο Serial Plotter, με και χωρίς το φίλτρο Kalman. Με μπλε χρώμα είναι η αναλογική τιμή όπως διαβάζεται από τους αισθητήρες κάμψης, ενώ με κόκκινο χρώμα είναι η αναλογική τιμή του ίδιου δαχτύλου έπειτα από την εφαρμογή του Kalman φίλτρου με την διαφορά να είναι εμφανής.

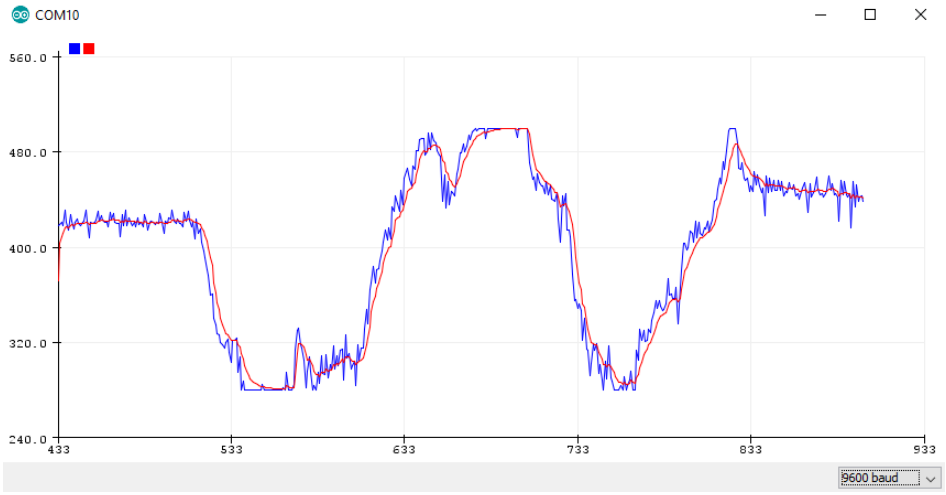
1) Αντίχειρας (Thumb)



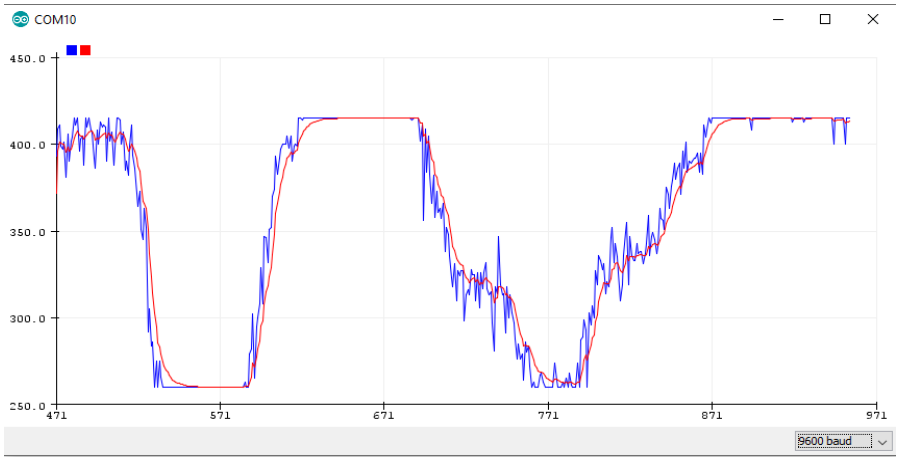
2) Δείκτης (Index)



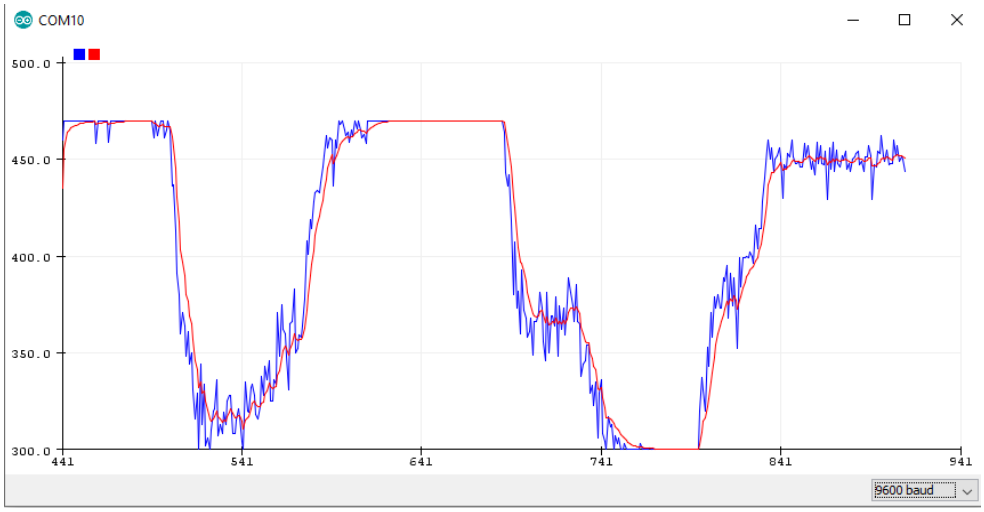
3) Μεσαίος (Middle)



4) Παράμεσος (Ring)



5) Μικρός (Pinky)



Αρχικοποιούμε τις νέες και παλιές εκτιμήσεις εντός της συνάρτησης UpdateEstimate και για τον πρώτο κύκλο εκτέλεσης οι νέες τιμές ισούνται με τις παλαιότερες και έπειτα ανανεώνονται κανονικά. Στον πίνακα current_estimate (και στον last_estimate αντίστοιχα) υποθέτουμε αρχικά ότι τα δάχτυλα θα βρίσκονται σε ευθεία γραμμή περίπου. Στον πίνακα current_estimate_error (και στον last_estimate_error) ορίζουμε την τιμή 400 που προκύπτει ως εξής: υποθέτουμε ότι η αρχική μας εκτίμηση θα έχει αρχικό σφάλμα περίπου 20 μονάδες (τυπική απόκλιση). Συνεπώς η μεταβλητή current_estimate_error (αντίστοιχα και η last_estimate_error) προκύπτει ως το τετράγωνο της τυπικής απόκλισης (διακύμανση) άρα current_estimate_error= $20^2 = 400$. Επίσης, οι αισθητήρες κάμψης έχουν ανοχή μέτρησης +-10% έως +-30% οπότε για το εύρος τιμών τάσης κάθε δαχτύλου που έχουμε (110-450), ορίζουμε ένα σταθερό σφάλμα μέτρησης (measure_error) = 40. Στη συνέχεια υπολογίζουμε το Kalman κέρδος (Gain), την εκτίμηση της τωρινής τιμής και έπειτα ανανεώνουμε τις αντίστοιχες μεταβλητές.

```
//kalman filter function
float UpdateEstimate(float finger_value, int n){
   //Αρχικοποίηση εκτιμήσεων για τις αρχικές θέσεις των δαχτύλων
  // όπου τωρινή εκτίμηση = παλιά εκτίμηση  για την πρώτη εκτέλεση
   static float last_estimate[5] = {380, 145, 345, 315, 355};
   static float current_estimate[5] = {380, 145, 345, 315, 355};
   static float last_estimate_error[5]= {400, 400, 400, 400, 400};
   static float current_estimate_error[5]={400, 400, 400, 400, 400};
   static const float q = 0.5;
   static const float measure_error = 40;
   //υπολογισμός Kalman gain για καθε δαχτυλο
   float kalman_gain = last_estimate_error[n]/(last_estimate_error[n] + measure_error);
   //εκτίμηση τωρινής τιμής
```

```
  current_estimate[n] = last_estimate[n] + kalman_gain * (finger_value - last_estimate[n]);

  //ανανέωση τωρινής αβεβαιότητας εκτίμησης

  current_estimate_error[n] = (1.0 - kalman_gain) * last_estimate_error[n] + fabs(last_estimate[n] - current_estimate[n])* q;

  //ανανέωση μεταβλητών

  last_estimate[n] = current_estimate[n];

  last_estimate_error[n] = current_estimate_error[n];

  return current_estimate[n];

}
```

Αφού φιλτράρουμε τις αναλογικές τιμές με το φίλτρο Kalman, φτιάχνουμε έναν πίνακα 6 στοιχείων όπου το πρώτο είναι το 0 (μηδέν) για να λειτουργεί ως δείκτης που σηματοδοτεί την μετάδοση του κάθε πακέτου με τις 5 τιμές των δαχτύλων. Έπειτα διαιρούμε τα στοιχεία όλα με το 2.35 για να σταλθούνε ως byte και όχι ως integer οι ανωτέρω τιμές, για να κάνουμε ταχύτερη τη σειριακή μετάδοση των πακέτων μας.

```
SendFingerValues[0] = 0;

for (int i=1; i<6; i++) {

        SendFingerValues[i] = FingersUpdatedValues[i-1] / 2.35;

}

for (int i=0; i<6; i++){

        BTserial.write(SendFingerValues[i]);     //εικονική σειριακή μέσω της βιβλιοθήκης //Software Serial για αποστολή των δεδομενων μέσω του HC05

        Serial.println(SendFingerValues[i] * 2.35);     //παρακολούθηση τιμών από την //hardware Serial (micro usb) μέσω του serial monitor

}
```

Τέλος, έχουν προστεθεί ένα πράσινο και ένα κόκκινο led που ενημερώνουν το χρήστη εάν η ασύρματη επικοινωνία μεταξύ των 2 Arduino (Uno – ρομποτικό χέρι και Nano - γάντι) μέσω των HC05, έχει διακοπεί ή όχι. Η διακοπή της επικοινωνίας έχει ως συνέπεια να ανάψει το κόκκινο led, ενώ σε περίπτωση που υπάρχει διαρκής επικοινωνία, το πράσινο led είναι αναμμένο. Αυτό έχει επιτευχθεί ως εξής: Κάθε φορά που το Arduino Uno (ρομποτικό χέρι) λαμβάνει έναν πίνακα με τις τιμές όλων των δαχτύλων και αφού τα επεξεργαστεί και δώσει τα σήματα ελέγχου στους σερβοκινητήρες, στέλνουμε ανά 500msec την τιμή 1 (σταθερά) μέσω του HC 05, πίσω στον Arduino Nano (Master) που βρίσκεται στο γάντι καθώς η Bluetooth επικοινωνία είναι αμφίδρομη μεταξύ ενός master και slave. Σε περίπτωση που δεν λάβει αυτή την τιμή για χρόνο μεγαλύτερο από 1 δευτερόλεπτο, τότε ανάβει το κόκκινο led.

4.2.6 Κώδικας Arduino Uno (Slave – ρομποτικό χέρι)

Αρχικοποίηση μεταβλητών

```
Unsigned long CurrentTime = 0;

Unsigned long PreviousCommunicationStatusSignalTime = 0;



void loop () {

//υπόλοιπο πρόγραμμα

CurrentTime  =millis();    //έναρξη συνεχούς μέτρησης

if  (CurrentTime - PreviousCommunicationStatusSignalTime>= 500){

        Serial.write(1); //(ΣΗΜΑ ΟΚ)

        PreviousCommunicationStatusSignalTime = CurrentTime;  //ανανέωση //μέτρησης

        }

}
```

4.2.7 Κώδικας Arduino Nano (Master – γάντι)

Αρχικοποίηση μεταβλητών

```
Const int GreenLED = 7;

Const int RedLED = 8;

unsigned long CurrentTime = 0;

unsigned long PreviousTime = 0;

const unsigned long TimeoutPeriod = 1000;

void loop() {

//υπόλοιπο πρόγραμμα

CurrentTime  =millis();

 if  (BTserial.available() > 0){

          byte Communication = BTserial.read();

           if  (Communication == 1){

                digitalWrite(GreenLED, HIGH);

                digitalWrite(RedLED, LOW);

            }

          previousTime = CurrentTime;

   }

 else if  (!BTserial.available() &&(CurrentTime - PreviousTime>= TimeoutPeriod)) {

          digitalWrite(GreenLED, LOW);

          digitalWrite(RedLED, HIGH);

}
```

Αφού συνοψίσαμε τη λειτουργία του γαντιού και τον τρόπο που στέλνει τα δεδομένα θα αναλύσουμε τον τρόπο που λαμβάνει τα δεδομένα ο δεύτερος ελεγκτής (Arduino Uno).

Προτού όμως εξετάσουμε την ανωτέρω λειτουργία πρέπει ο Arduino Uno να στέλνει τα κατάλληλα σήματα ελέγχου στον PCA 9685 driver.

4.2.8 PCA 9685 servo driver

Η σύνδεση του PCA9685 με τον Arduino γίνεται ως εξής:

Pins PCA9685

- VCC όπου δίνουμε τα 5V από τον Arduino. Δεν τροφοδοτεί με 5V τους σερβοκινητήρες, παρά μόνο την πλακέτα.
- GND -> GND του Arduino
- SDA ->Analog 4 του Arduino
- SCL ->Analog 5 του Arduino

Ουσιαστικά τα pins SCL και SDA, είναι Pin ρολογιού (clock) και pin δεδομένων (data) αντίστοιχα στο I2C πρωτόκολλο επικοινωνίας. Μέσω του τροφοδοτικού δίνουμε τάση 5V στα Pins power που τροφοδοτεί με ρεύμα τους συνδεδεμένους σερβοκινητήρες.

Κατεβάζουμε τη βιβλιοθήκη **Adafruit_PWMServoDriver** μέσω του library manager και δημιουργούμε ένα αντικείμενο με ονομασία pwm.

```
#include<Wire.h>

#include <Adafruit_PWMServoDriver.h>

Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();

#define  SERVOMIN  150 // ελάχιστη διάρκεια παλμού

#define  SERVOMAX  500 // μέγιστη διάρκεια παλμού

#define USMIN  600

#define  USMAX  2400

#define  SERVO_FREQ 50 // Συχνότητα σερβοκινητήρων στα 50Hz
```

Η εντολή για την κίνηση ενός σερβοκινητήρα για τη συγκεκριμένη βιβλιοθήκη είναι:

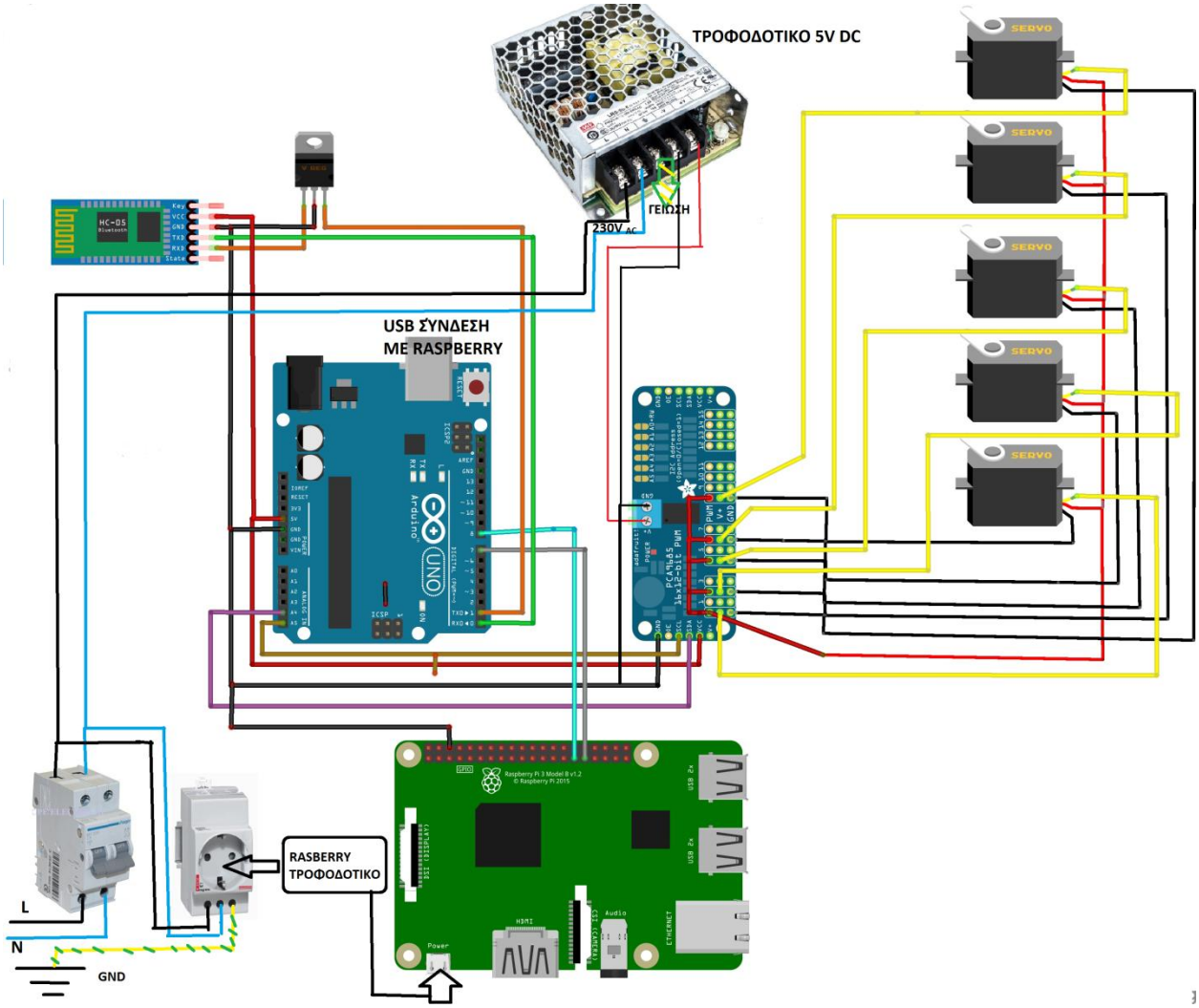
- pwm.setPWM (σερβοκινητήρας στο pin 0 έως 15, 0, ServoPosition) όπου το πρώτο όρισμα είναι το κανάλι, το δεύτερο και τρίτο είναι η διάρκεια που ο παλμός του PWM θα είναι σε κατάσταση ON (high) από 0 έως την τιμή του ServoPosition και έπειτα θα γίνει OFF (low) από την τιμή του ServoPosition +1 έως το 4095.

Πειραματικά βρέθηκε ότι για την σύμπτυξη ενός δαχτύλου του ρομποτικού χεριού η διάρκεια του pwm σήματος ServoPosition είναι 150, ενώ για την έκταση είναι από 450-510 που αντιστοιχεί σε πλάτος παλμού περίπου 0.75 msec και 2.25-2.5 msec αντίστοιχα.

4.2.9 Arduino Uno Receiver (Slave)

Συνοψίζοντας τη λειτουργία όλων των ανωτέρω επιμέρους εξαρτημάτων δημιουργούμε το παρακάτω ηλεκτρολογικό κύκλωμα. Η τροφοδοσία όλου του συστήματος γίνεται με πρίζα 230V που περνάει από έναν διπολικό ασφαλειοδιακόπτη εντάσεως 10A, που δίνει ισχύ σε μια πρίζα ράγας και σε ένα τροφοδοτικό $5V_{DC}$ εντάσεως 5A. Στην πρίζα ράγας τοποθετήθηκε το τροφοδοτικό των $5V_{DC}$ του Raspberry Pi,το οποίο μέσω ενός καλωδίου USB (Type A αρσενικό σε Type B αρσενικό) τροφοδοτεί με 5Vτο Arduino Uno της κατασκευής μας. Το τροφοδοτικό των 5V - 5A, δίνει ρεύμα σε όλα τους 5 σερβοκινητήρες μέσω του PCA9685 ενώ για τη σύνδεση του σήματος Rx του HC05 bluetooth module με τον Arduino Uno, χρησιμοποιήθηκε ένας ρυθμιστής τάσης (voltage regulator) από 5 σε 3.3V.

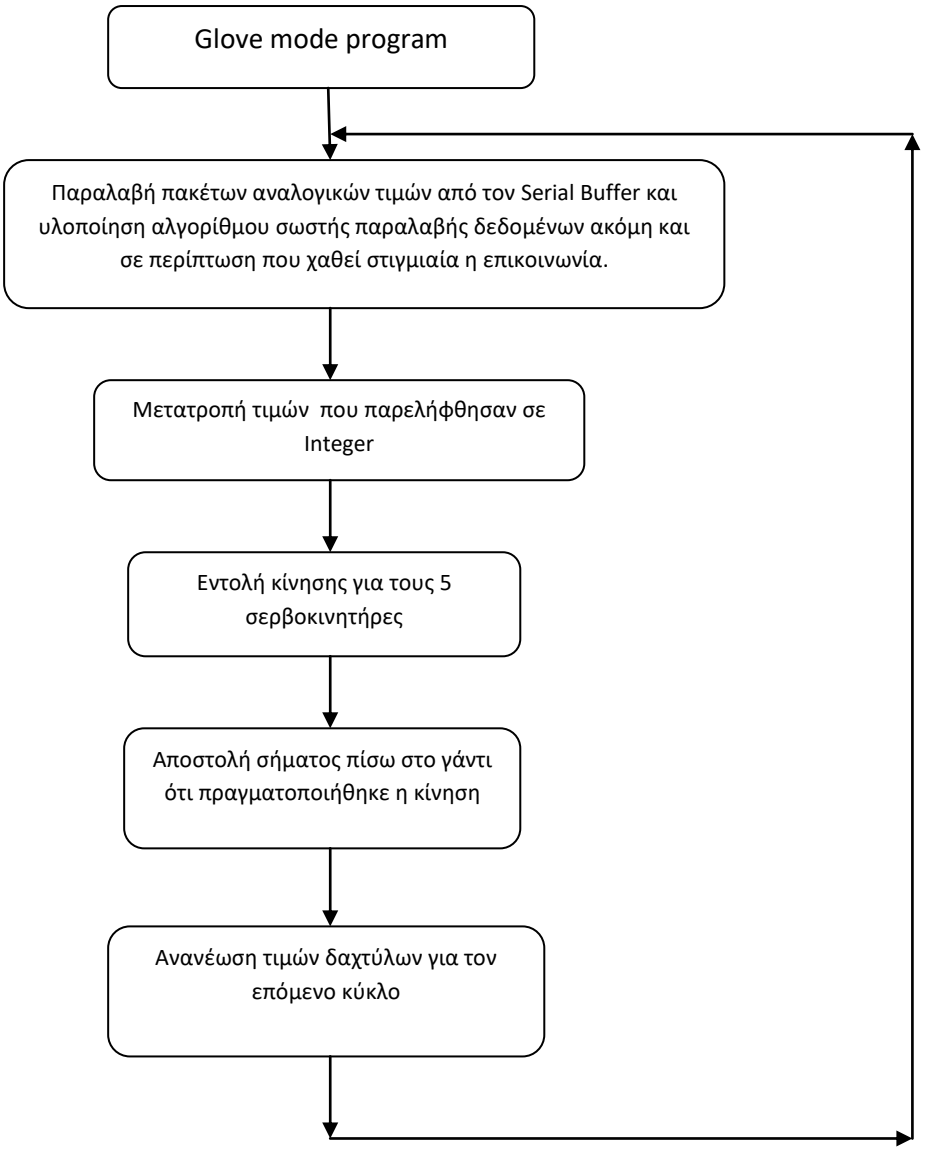
Τέλος, θα πρέπει να σημειωθεί ότι είναι απαραίτητη η σύνδεση όλων των 0V σημείων για τη λειτουργία του κυκλώματος.



Εικόνα 40: Συνδεσμολογία για ρομποτικό χέρι βάσης

Όταν ο χρήστης γυρίσει τον επιλογικό διακόπτη στη θέση 2, τότε διαβάζει το Raspberry την κατάστασή του και αντίστοιχα ενεργοποιεί την έξοδο 6 (BCM) που συνδέεται στην είσοδο 8 του Arduino Uno.

Ακολουθεί το διάγραμμα ροής για το glove_mode function που ελέγχει το ρομποτικό χέρι μέσω του PCA 9685 driver.



Γράφημα 6: Λογικό διάγραμμα βάσης για τον έλεγχο του ρομποτικού χεριού μέσω γαντιού

4.2.10 Κώδικας Arduino Uno

```
void setup() {

        Serial.begin(38400);

        pwm.begin();

        pwm.setOscillatorFrequency(27000000);

        pwm.setPWMFreq(SERVO_FREQ);  // ~50 Hz συχνότηταservos

        delay(10);

        pinMode(CameraMode, INPUT);    //Pin7
```

```
        pinMode(GloveMode, INPUT);  // Pin8
//κλεισιμο ρομποτικου χεριου
        PwmSignals(PulsesArray,  0);   //1η σειρά του 2d πίνακα Pulses Array που είναι η
//κλειστή θέση
        delay(2000);
}
//-------------------------------------------------------------
void loop() {
if (digitalRead(CameraMode) == HIGH) {
        camera_mode();
  }
else if (digitalRead(GloveMode) == HIGH) {
        glove_mode();
  }

else {
   //κλειστη θεση χεριου
   PwmSignals(PulsesArray, 0);  //1η σειρα του 2d πινακα Pulses Array
   }
}
```

Αξίζει να σημειωθεί ότι η πλευρά του γαντιού στέλνει συνεχώς 6 τιμές, όπου η πρώτη είναι το μηδέν ως δείκτης αρχής του εκάστοτε πακέτου, ακολουθούμενου από τις 5 τιμές για κάθε δάχτυλο. Σε περίπτωση που ο Arduino Uno στο ρομποτικό χέρι ο οποίος λαμβάνει τα πακέτα

σειριακά μέσω του HC05 bluetooth, χάσει την επικοινωνία ή δεν λάβει κάποια από τις 6 παραπάνω τιμές στη σωστή σειρά, η κίνηση του ρομποτικού χεριού θα είναι λανθασμένη. Για αυτό το λόγο φτιάχνουμε έναν πίνακα 12 byte (IncomingData) και ελέγχουμε ότι στον Serial Buffer του ελεγκτή καταφθάσανε αυτά τα 12 byte μέσω της Serial.available εντολής.

Πώς λειτουργεί όμως η εντολή Serial.available σε συνδυασμό με την Serial.read; Ο serial Buffer του Arduino Uno χωράει έως 64 byte που έρχονται σειριακά, και αφού περιμένουμε να γεμίσει 12 byte ο Buffer, στη συνέχεια διαβάζουμε μέσω της εντολής Serial.read ένα byte τη φορά και στη συνέχεια το αφαιρεί από τον Buffer. Ο λόγος που επιλέξαμε να ελεγχθεί εάν έχουν φτάσει 12 byte στον Buffer είναι ο εξής: Όταν ανιχνευθεί το μηδέν (δείκτης) σε οποιαδήποτε θέση για πρώτη φορά, τότε αποθηκεύουμε τη θέση του μηδέν στο εκάστοτε πακέτο των 12 byte και γνωρίζοντας ότι ακολουθείτε από 5 byte τιμές των δαχτύλων, αποθηκεύουμε σε νέο πίνακα 5 θέσεων αυτές τις τιμές. Το μέγεθος του Buffer πρέπει να είναι της τάξης των 6 byte το λιγότερο. Σε περίπτωση που η επικοινωνία μεταξύ του master και του slave δεν χανότανε ποτέ από τη στιγμή που ο slave ξεκινούσε να λαμβάνει τιμές από τον master (γάντι), τότε ο Buffer (Serial.available) θα είχε μέγεθος 6 byte, ένα για τον δείκτη 0 ακολουθούμενος από 5 byte που αντιπροσωπεύουν την αναλογική τιμή του αισθητήρα κάμψης του κάθε δαχτύλου. Σε περίπτωση που χαθεί η επικοινωνία μεταξύ master και slave, το χειρότερο σενάριο είναι ο δείκτης 0 στο εκάστοτε πακέτο των 6 byte, να είναι στη θέση 6 (όπως φαίνεται και στον πίνακα 7) συνεπώς ο Serial Buffer πρέπει να έχει μέγεθος τουλάχιστον 11byte.

0 (πρώτο πακέτο)	1ο byte	2ο byte	3ο byte	4ο byte	5ο byte	0 (δεύτερο πακέτο)	1ο byte	2ο byte	3ο byte	4ο byte	5ο byte

Πίνακας 6: Πρώτη περίπτωση χωρίς διακοπή επικοινωνίας μεταξύ master-slave

1ο byte	2ο byte	3ο byte	4ο byte	5ο byte	0 (πρώτο πακέτο)	1ο byte	2ο byte	3ο byte	4ο byte	5ο byte	0 (δεύτερο πακέτο)

Πίνακας 7: Δεύτερη περίπτωση όπου έχει χαθεί η επικοινωνία μεταξύ master-slave

```
CurrentTime  = millis();

 if (Serial.available()>= 12) {

   Counter = 0;    //flag variable.

   for (int i =0; i<12;  i++){

         IncomingData[i] = Serial.read();

         If  (IncomingData[i] == 0  && Counter <  1){       /

               Counter  +=  1;          //το πρώτο 0 που θα συναντήσουμε στο πακέτο των 12
//αριθμων του Buffer απο το serial read το αποθηκεύουμε στη μεταβλητή ZeroPos

               ZeroPos = i;

           }

    }
```

Έπειτα οι τιμές των δαχτύλων προκύπτουν με τον πολλαπλασιασμό με τη σταθερά 2.35 (αντίθετα με ότι κάναμε στον Arduino Nano για να στείλουμε ως byte τις τιμές με τη διαίρεση με το 2.35).

```
for (int j =0; j<5; j++){

     fingers[j] = IncomingData[j + ZeroPos + 1] * 2.35;

}
```

Αφού λάβαμε τις τιμές για τα δάχτυλα, καλούμε τη συνάρτηση ServoMotion (για κάθε δάχτυλο).

```
for (int i=0; i<5; i++){

   ServoMotion(CurrentFingersValues[i],  PreviousFingersValues[i],  SensorLimits[i*2  +1],
SensorLimits[i*2], ServoPulseLength[i*2], ServoPulseLength[i*2 +1], i);

}
```

```
//servo motion function

Void ServoMotion(int CurrentFingerValue, int PreviousFingerValue, int SensorLimit1, int SensorLimit2, int PulseLengthLow, int PulseLengthHigh, int Counter) {

   if (CurrentFingerValue>= PreviousFingerValue){

         for (int i = PreviousFingerValue; i <= CurrentFingerValue; i++) {

              intServoPosition = i;

              ServoPosition = map(ServoPosition, SensorLimit1, SensorLimit2, PulseLengthLow, PulseLengthHigh);

              pwm.setPWM(Counter*2, 0, ServoPosition);

         }

     }

else {

         for (int i = PreviousFingerValue; i >= CurrentFingerValue; i--) {

              intServoPosition = i;

              ServoPosition = map(ServoPosition, SensorLimit1, SensorLimit2, PulseLengthLow, PulseLengthHigh);

               pwm.setPWM(Counter*2, 0, ServoPosition);

           }

      }

}
```

Στη συνάρτηση αυτή αρχικά ελέγχουμε εάν η τωρινή αναλογική τιμή του δαχτύλου είναι μεγαλύτερη από την προηγούμενη τιμή. Αυτό σημαίνει ότι το δάχτυλο κάνει έκταση. Σε διαφορετική περίπτωση κάνει κάμψη. Η ServoPosition μεταβλητή δέχεται γραμμικοποίηση από τα όρια των SensorLimit(1 και 2) στην τιμή PulseLength (Low και High) μέσω του built

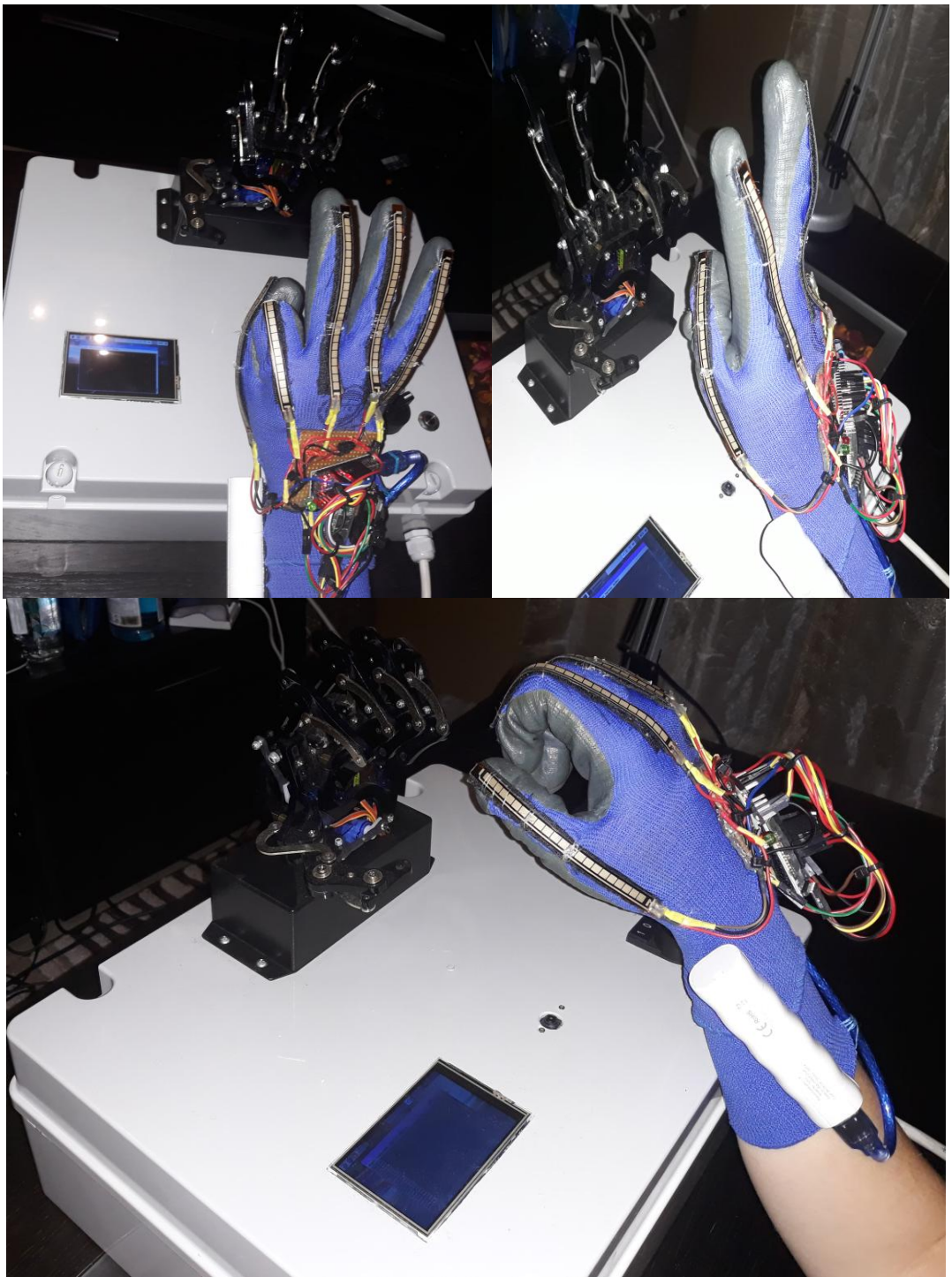
in function map. Οι τιμές των μεταβλητών SensorLimit1 και 2 είναι οι ελάχιστες και μέγιστες τιμές που αντιστοιχούν στις αναλογικές τιμές των αισθητήρων κάμψης (Πίνακας 5). Οι τιμές PulseLength είναι οι τιμές των παλμών που αντιστοιχούνε στην κίνηση των servοκινητήρων. Τα ανωτέρω συνοψίζονται και στον ακόλουθο πίνακα.

Δάχτυλο	**SensorLimit1 (πλήρη κάμψη)**	**PulseLengthLow**	**SensorLimit2 (πλήρη έκταση)**	**PulseLengthHigh**
Αντίχειρας (thumb)	305	150	450	500
Δείκτης (index)	110	150	185	505
Μεσαίος (middle)	238	160	445	495
Παράμεσος (Ring)	235	150	400	510
Μικρός (Pinky)	260	165	453	495

Πίνακας 8: Πίνακας αντιστοίχισης τιμών αισθητήρων κάμψης με τιμές παλμών για την κίνηση των σερβοκινητήρων

Τέλος, ανανεώνουμε τις τιμές των δαχτύλων για την επόμενη παραλαβή των πακέτων.

```
for (int i=0; i<5; i++) {

        PreviousFingersValues[i] = CurrentFingersValues[i];

}
```



Εικόνα 41: Έλεγχος ρομποτικού χεριού με απτικό γάντι μέσω bluetooth

4.3 Αναγνώριση δαχτύλων και κίνηση ρομποτικού χεριού μέσω κάμερας και Raspberry Pi (2ο mode ελέγχου)

Με τη ρομποτική να εισχωρεί διαρκώς περισσότερο στη ζωή μας, είναι επιθυμητό πολλές φορές να υπάρχει κάμερα η οποία θα αντιλαμβάνεται κινήσεις ή χειρονομίες του ανθρώπου και θα εκτελεί συγκεκριμένες λειτουργίες [B8]. Τέτοιες εφαρμογές συναντώνται στην ιατρική, στη νοηματική, στα έξυπνα σπίτια, στη βιομηχανία, στα αυτοκίνητα κτλ.

Σε αυτό το mode ελέγχου που επιλέγεται γυρίζοντας τον επιλογικό διακόπτη στα αριστερά, μια κάμερα συνδέεται με Raspberry Pi και κάνει αναγνώριση του δεξιού χεριού του χρήστη, και αντίστοιχα κινείται και το ρομποτικό χέρι. Χρησιμοποιήθηκε Raspberry Pi 3B+ το οποίο επικοινωνεί με τον Arduino Uno (μέσω USB και σειριακής UART επικοινωνίας) το οποίο ελέγχει τους σερβοκινητήρες του ρομποτικού χεριού μέσω του PCA9685 servo driver. Για την απεικόνιση χρησιμοποιήθηκε μια Lcd 3.5'' που κουμπώνει στα pins του Raspberry ενώ η κάμερα (pi camera v2) κουμπώνει μέσω καλωδιοταινίας απευθείας στην CSI του Raspberry.

4.3.1 Εγκατάσταση της OpenCV

Δημιουργήσαμε ένα virtual environment από όπου τρέχουμε το python script. Το virtual environment δημιουργήθηκε για τον εξής λόγο:

Το εικονικό περιβάλλον (virtual environment ή εν συντομία venv), δημιουργεί ένα ξεχωριστό περιβάλλον για τα διάφορα προγράμματα που μπορεί να τρέξει ένας χρήστης, ενώ παράλληλα μπορεί εντός του, να υπάρχουν διαφορετικές εκδόσεις π.χ της Python ή ακόμα και βιβλιοθήκες χωρίς να επηρεάζεται από το γενικό περιβάλλον (global environment) όπου είναι π.χ εγκατεστημένη η Python, δημιουργώντας συνεπώς ανεξάρτητα projects.

Αρχικά πρέπει να γίνει η εγκατάσταση της OpenCV 4 στο Raspberry. Η εγκατάσταση ενός ''πακέτου'' (package) σε Linux λειτουργικό σύστημα, μπορεί να γίνει με 2 τρόπους. Με την εγκατάσταση ενός προ-χτισμένου πακέτου (pre built package) ή να κάνει τη μεταγλώττιση από τον πηγαίο κώδικα (compile from source) ο χρήστης. Το πλεονέκτημα του πρώτου τρόπου είναι η γρήγορη και εύκολη εγκατάσταση του πακέτου και των αντίστοιχων dependencies αυτού (μέσω του package manager), αλλά κάποιες λειτουργίες μπορεί να μην είναι βελτιστοποιημένες (optimized) για το σύστημά μας ή ακόμα και να εμφανιστούνε σφάλματα σε κάποιες λειτουργίες του. Με τον δεύτερο τρόπο έχουμε τα εξής οφέλη: βελτιστοποιημένη έκδοση του πακέτου για το σύστημά μας, ενδεχομένως νεότερη έκδοση

του πακέτου και δυνατότητα να επιλέξουμε ή όχι συγκεκριμένες λειτουργίες του πακέτου. Στη συγκεκριμένη περίπτωση επιλέξαμε το δεύτερο τρόπο εγκατάστασης της OpenCV 4.0.0 εντός ενός εικονικού περιβάλλοντος (virtual environment) που δημιουργήσαμε με όνομα cv.

Τέλος, για να ελέγξουμε ότι έγινε σωστή εγκατάσταση της OpenCV εκτελούμε τις ακόλουθες εντολές στο command line:

Φορτώνουμε το profile αρχείο.

$ source ~/.profile

Φορτώνουμε το virtual environment (venv) που δημιουργήσαμε για την OpenCV. Εάν στην έκδοση της OpenCV μας επιστρέψει την έκδοση τότε η εγκατάσταση ήταν επιτυχής.

```
$ workon cv
$ python
>>>import cv2
>>> cv2.__version__
'4.0.0'
>>>exit()
```

4.3.2 Τρόπος προσέγγισης

Η μέθοδος που επιλέξαμε ονομάζεται background subtraction όπου αρχικά η κάμερα τραβάει φωτογραφία του χώρου χωρίς το χέρι στο τοπίο και έπειτα κάθε επόμενο frame συγκρίνεται με την αρχική εικόνα και αφού γίνουν οι απαραίτητες ενέργειες (Gaussian φίλτρο για μείωση θορύβου και κατωφλίωση-thresholding) γίνεται απόλυτη αφαίρεση και προκύπτει το χέρι (ως λευκό) ενώ το φόντο (background)είναι μαύρο. Επειδή η κάμερα δεν προλαβαίνει να προσαρμοστεί στον φωτισμό, η αρχική φωτογραφία τραβιέται αφού περάσουν π.χ 30 έως 60 frames.

Αφού εντοπιστεί το χέρι εντός του γαλάζιου πλαισίου (bounding box) που ορίζουμε, ξεκινάει ο αλγόριθμος εντοπισμού των δαχτύλων όπου βρίσκουμε τις θέσεις των δαχτύλων καθώς και τις θέσεις με τις καμπυλότητες μεταξύ των δαχτύλων.

Η αναγνώριση των δαχτύλων γίνεται με ευκλείδιες αποστάσεις, συγκρίσεις και συγκρίσεις ως προς το κέντρο της παλάμης σε γενικές γραμμές.

Οι περιπτώσεις που επιλέχθηκαν για την αναγνώριση είναι οι εξής:

Αριθμός δαχτύλων	**(Υπο)Περιπτώσεις και κωδικοποίηση για σειριακή επικοινωνία**
0	Κλειστή παλάμη (‘**A**’)
1	Δείκτης (‘**B**’)
2	Α) αντίχειρας, δείκτης(‘**C**’) Β) δείκτης, μεσαίος (‘**D**’) Γ) δείκτης, μικρός (‘**E**’)
3	Α) αντίχειρας, δείκτης, μεσαίος (‘**F**’) Β) αντίχειρας, δείκτης, μικρός (‘**G**’) Γ) δείκτης, μεσαίος, παράμεσος (‘**H**’) Δ) μεσαίος, παράμεσος, μικρός (‘**I**’)
4	Α) αντίχειρας, δείκτης, μεσαίος, μικρός(‘**J**’) Β) αντίχειρας, μεσαίος, παράμεσος, μικρός (‘**K**’) Γ) δείκτης, μεσαίος, παράμεσος, μικρός (‘**L**’)
5	Ανοικτή παλάμη (‘**M**’)

Πίνακας 9: Πίνακας περιπτώσεων

Αφού γίνει η αναγνώριση, στέλνουμε για κάθε περίπτωση από τις παραπάνω, έναν χαρακτήρα (από Α έως Μ για τις 13 ανωτέρω περιπτώσεις) σειριακά στον Arduino Uno (μέσω USB) όπου στέλνει τα κατάλληλα σήματα στο PCA9685 για την οδήγηση των σερβοκινητήρων.

Στην Lcd οθόνη εμφανίζονται ο αριθμός των δαχτύλων ενώ από κάτω εμφανίζονται τα ονόματα των δαχτύλων που εντοπίζονται καθώς και μήνυμα όταν δεν εντοπίζεται χέρι εντός του πλαισίου (no contours detected).

Γενικώς, παρατηρήθηκε ότι ο αλγόριθμος και η κάμερα επηρεάζονται από τα επίπεδα φωτεινότητας καθώς και από τις σκιάσεις (π.χ ενώ υπάρχει πηγή φωτός σε σκοτεινό περιβάλλον) ενώ επίσης και το φόντο μπορεί να επηρεάσει τα αποτελέσματα (π.χ γωνίες και περιβάλλον με μεγάλα αντικείμενα εντός του frame).

4.3.3 Υλοποίηση

Για αυτό το mode ελέγχου, πρέπει μέσω κάμερας που συνδέεται απ’ ευθείας στη CSI θύρα του Raspberry, να αναγνωρίσουμε πόσα δάχτυλα ‘’βλέπει’’ η κάμερα καθώς και ποια είναι αυτά ώστε έπειτα να στείλουμε τα κατάλληλα σήματα στον Arduino Uno που θα κινήσει κατάλληλα τους σερβοκινητήρες των δαχτύλων του ρομποτικού χεριού.

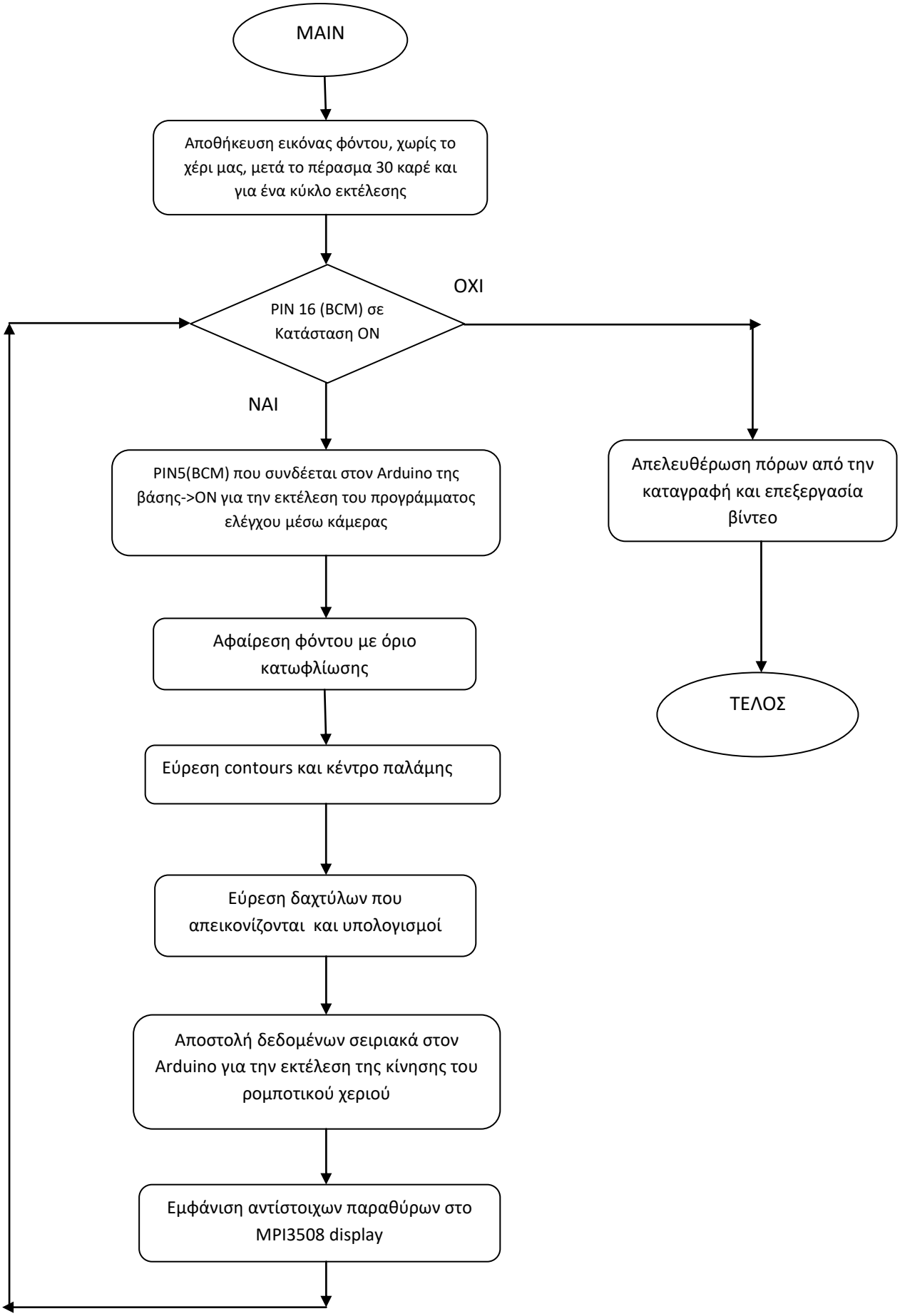
Στάδια:

1) Αρχικά θέλουμε να απομονώσουμε το περίγραμμα του χεριού από το βίντεο. Φυσικά δεν πρέπει να ξεχνάμε ότι ένα βίντεο είναι συνεχείς λήψεις εικόνων, συνεπώς όταν εκτελεστεί ο αλγόριθμος επεξεργαζόμαστε κάθε frame του video που καταγράφουμε. Η κάμερα στην αρχή κάνει λήψη για 60 καρέ μέχρι να σταθεροποιηθεί η εικόνα (warming up) και έπειτα αποθηκεύουμε αυτήν την εικόνα ως την πρώτη εικόνα του χώρου (first image) αφού πρώτα την μετατρέψουμε σε Grayscale και εφαρμόσουμε **Gaussian φίλτρο**. Έπειτα, για κάθε επόμενη εικόνα που λαμβάνουμε εφαρμόζουμε εκ νέου μετατροπή σε Grayscale και Gaussian φιλτράρισμα. Από την απόλυτη αφαίρεσή τους και με ένα όριο κατωφλίωσης που επιλέγουμε εμείς προκύπτει ως άσπρο οτιδήποτε προστεθεί μπροστά στην κάμερα. **Η μέθοδος αυτή ενδείκνυται για εφαρμογές όπου η κάμερα είναι στατική**.

2) Ελέγχουμε εάν υπάρχει το χέρι στο φόντο. Εάν υπάρχουν 2 ξεχωριστά αντικείμενα (επιφάνειες ανεξάρτητες) εντός του frame επεξεργασίας, υποθέτουμε ότι το μεγαλύτερο από αυτά είναι το χέρι μας. Βρίσκουμε επίσης και το κέντρο της επιφάνειας του χεριού ως προς τους άξονες χ και y (image moments). Σύμφωνα με κάποιες συναρτήσεις της OpenCV καθώς και γεωμετρικούς υπολογισμούς και συγκρίσεις βρίσκουμε ποια δάχτυλα είναι εντός του frame.

3) Αποστολή σειριακά των δεδομένων για κάθε μια από τις 13 περιπτώσεις του πίνακα (πίνακας με περιπτώσεις δαχτύλων) στον Arduino της βάσης.

4) Εμφάνιση 2 παραθύρων στην οθόνη MPI3508. Τα παράθυρα αυτά είναι η εικόνα που προκύπτει το χέρι μας ως άσπρη επιφάνεια με τη μέθοδο της αφαίρεσης του φόντου (background subtraction), ενώ το δεύτερο απεικονίζει ότι βλέπει η κάμερα μαζί με μηνύματα όπως το πόσα και ποια δάχτυλα αναγνωρίστηκαν, καθώς και μήνυμα που ειδοποιεί το χρήστη ότι η θερμοκρασία του επεξεργαστή του Raspberry εάν αυτή ξεπεράσει τους 85°C, καθώς και το πλαίσιο όπου ο χρήστης πρέπει να τοποθετήσει το χέρι του για την αναγνώριση (bounding box) .



Γράφημα 7: Γενικό διάγραμμα ροής για τον έλεγχο μέσω κάμερας

Για το python script αρχικά εισάγουμε τις κατάλληλες βιβλιοθήκες:

```
import cv2

import imutils

import numpy as np

import math

import time

import serial

import sys

import RPi.GPIO as GPIO

from  scipy.spatial import distance
```

Και ορίζουμε ότι η USB θύρα θα στέλνει σειριακά δεδομένα με baudrate 38400bps.

Για να βρούμε σε ποια USB θύρα επικοινωνεί με τον Arduino γράφουμε στο command window την εντολή: ls /dev/tty*

και έπειτα δηλώνουμε το όνομα της θύρα αυτής στο πρόγραμμά μας.

```
ser = serial.Serial('/dev/ttyACM0', 38400)

ser.flush()
```

Ρυθμίζουμε τα GPIO pins με αρίθμηση BCM, και επιλέγουμε την ενσωματωμένη pull-down αντίσταση.

```
GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM)

GPIO.setup(16, GPIO.IN, pull_up_down=GPIO.PUD_UP)  #pin εκτελει το script. εαν δεν #ειναι συνδεδεμενο, η εξοδος δινει 0v. εαν ειναι συνδεδεμενο στελνει 3.3v #στο arduino

GPIO.setup(26, GPIO.IN, pull_up_down=GPIO.PUD_UP)  #pin gia ektelesi glove control
```

```
GPIO.setup(5, GPIO.OUT)  #εξοδος ελεγχου του arduino camera mode

GPIO.setup(6, GPIO.OUT)  #εξοδος ελεγχου του arduino glove mode

GPIO.setup(23, GPIO.OUT)  #εξοδος για ελεγχο relay για serial com to pins 0,1

input_state = GPIO.input(16)

input_state2 = GPIO.input(26)

GPIO.output(5, 0)

GPIO.output(6, 0)

GPIO.output(23, 0)
```

Ρυθμίζουμε επιπλέον τις θέσεις και το μέγεθος των μηνυμάτων στο MPI3508 display καθώς και τα όρια για το bounding box.

```
font = cv2.FONT_HERSHEY_SIMPLEX

text_pos = (100, 35)      #no contours text position

text_pos2 = (150, 420)    #position for fingers are:

text_pos3 = (150, 450)   # position for 2 fingers text

text_pos4 = (125, 450)   #position for 3 fingers text

text_pos5 = (60, 450)   #position for 4 fingers text

text_pos6 = (150,480)

text_pos7 = (100, 240)       #high cpu temp message

fontScale2 = 2

fontScale = 1

text_color = (10, 30, 250)  #color του text

text_thickness = 2
```

```
top, right, bottom, left = 70, 80, 450, 570     #bounding box dimensions όπου θα τοποθετεί το
#χέρι του ο χρήστης
```

Επειδή η OpenCV σε video είναι απαιτητική από πόρους, αυτό αυξάνει τη θερμοκρασία του raspberry και όταν ξεπεράσει τους 85°C ρίχνει τη συχνότητα εκτέλεσης του προγράμματος για να μην υπερθερμανθεί ο επεξεργαστής αλλά με αποτέλεσμα να γίνει πολύ αργή ή ακόμη και σχεδόν αδύνατη η εκτέλεση του προγράμματος (throttling). Συνεπώς στο πρόγραμμά μας, διαβάζουμε τη θερμοκρασία της cpu και αφού τη μετατρέψουμε σε List όπου παίρνουμε μόνο τα 4 στοιχεία για τη θερμοκρασία (επιστρέφει και πληροφορίες που δεν χρειαζόμαστε), ελέγχουμε εάν ξεπεράσει συνεχώς τους 80.5°C και εμφανίζουμε μήνυμα στην οθόνη για τον χρήστη ώστε εάν χρειαστεί να διακόψει την εκτέλεση του προγράμματος. Σε περίπτωση που ξεπεράσει τους 85°C τότε τερματίζουμε την εκτέλεση του script και εμφανίζουμε αντίστοιχο μήνυμα.

Σημείωση: Στην Python, σε script, η σειρά είναι η εξής: πρώτα εμφανίζονται οι βιβλιοθήκες που εισάγονται, Global μεταβλητές, functions και στο τέλος η main. Για λόγους όμως κατανόησης, τοποθετούμε εδώ πρώτα την main και έπειτα τα εκάστοτε functions. Στη main έχουμε:

```
#---------------M A I N ---------------------------------------------------------------

while True :

    first_scan = True

    input_state = GPIO.input(16)

    input_state2 = GPIO.input(26)

    # με false όταν το pin16 κλεινει κυκλωμα

    if (input_state == False) and (first_scan == True) :  #εκτελεση για 1η φορά μέχρι να
#πάρουμε την εικόνα του φόντου χωρίς το χέρι μας.

        GPIO.output(23, 0) #εξοδος για serial com OFF

        cap, first_frame, roi_first_frame, first_gray = first_image_capture()
```

```
    while (input_state == False) :    #οσο το Pin είναι ενεργο (επιλεχθηκε αυτο το mode #λειτουργιας
        print('camera mode')
        GPIO.output(23, 0) #εξοδος για serial com OFF
        first_scan = False
        input_state = GPIO.input(16)
        GPIO.output(5, 1)  #output for arduino camera mode on.
        GPIO.output(6, 0)  #output for arduino glove mode off.
        #καλούμε το function continuous capture για την επεξεργασια του κάθε frame
        frame, difference, roi_difference, intensity = continuous_capture()
        #καλούμε το function contours calc για την ανίχνευση του χεριου
         contours_calc(frame, intensity, difference)    # το function αυτο καλει  εσωτερικα και #την fingers_calc για την αναγνωριση των δαχτυλων
        #καλουμε το function show windows μονο για την εμφανιση των παραθυρων στην #Lcd οθόνη
         show_windows(first_frame, frame, roi_difference)
        key = cv2.waitKey(30)   #για εμφανιση των παραθυρων
        if (input_state == True) :     #εαν γυρισουμε το διακοπτη σε άλλο mode
                break
    if (input_state2 == False) :   #επιλογικος διακοπτης στη θεση 2 δεξια
        print('glove mode')
        GPIO.output(5, 0)  #camera mode εξοδος off
        GPIO.output(6, 1)  #glove mode εξοδος on
```

```
        GPIO.output(23, 1)  #εξοδος για ελεγχο relay serial com  ON
    else :
        GPIO.output(5, 0)  #camera mode εξοδος  off
        GPIO.output(6, 0)  #glove mode εξοδος  off
        GPIO.output(23, 0)  # εξοδος για ελεγχο relay serial com  ON


    try:
        cap.release()
        cv2.destroyAllWindows()
    except  NameError as error1:
         pass
    key = cv2.waitKey(30)
    if (key == 27) :
        GPIO.output(5, 0)   # εξοδος 5 OFF
        GPIO.output(6, 0)  #εξοδος 6 OFF
        break
GPIO.cleanup()
```

Αφού εμφανίσουμε το χέρι μέσω της μεθόδου της αφαίρεσης του φόντου, καλούμε τη συνάρτηση contours_calc(frame, intensity, difference) όπου αυτή με τη σειρά της καλεί εσωτερικά και την fingers_calc συνάρτηση. Η συνάρτηση contours_calc επιτελεί την εξής λειτουργία:

Δέχεται ως ορίσματα το εκάστοτε frame από την κάμερα, την φωτεινότητα των pixels κατά χ και y άξονες καθώς και **την εικόνα που προκύπτει από την αφαίρεση του φόντου (difference) με το εκάστοτε frame που επεξεργαζόμαστε**.

Η OpenCV μας διαθέτει μερικά πολύ χρήσιμα functions όπως το findContours με κυρίως όρισμα την εικόνα **difference** και επιστρέφει τα σημεία x, y όπου η φωτεινότητα ενός αντικειμένου αλλάζει σημαντικά, δηλαδή μας δίνει τα σημεία του περιγράμματος του αντικειμένου που ψάχνουμε. Εάν δεν έχουμε επιφάνεια (contour) να επιστρέφεται από τη συνάρτηση findContours τότε εμφανίζουμε μήνυμα ότι δεν υπάρχει χέρι μπροστά από την κάμερα, αλλιώς εάν η φωτεινότητα είναι εντός κάποιων ορίων τότε υπάρχει χέρι εντός του frame.

Είναι όμως πιθανό να εμφανιστεί στην κάμερα και δεύτερο περίγραμμα, όπου τότε ορίζουμε ότι η μεγαλύτερη ανεξάρτητη επιφάνεια είναι το χέρι μας.

Ένα σημείο προσοχής είναι και το εξής: Επειδή τα σημεία που μας επιστρέφει η συνάρτηση findContours είναι πολλά και θα επιβαρύνει τους μετέπειτα υπολογισμούς και τη μνήμη, καλούμε τη συνάρτηση **cv2.arcLength και cv2.approxPolyDP** [B3]. Η πρώτη συνάρτηση, (cv2.arcLength) και μας δίνει την περίμετρο της επιφάνειας του χεριού (contour). Έπειτα, χρησιμοποιώντας έναν συντελεστή, τον πολλαπλασιάζουμε με τον αριθμό που προκύπτει από την περίμετρο του χεριού, και το δίνουμε ως όρισμα στη δεύτερη συνάρτηση **(cv2.approxPolyDP)**, μαζί με την αρχική contour επιφάνεια. Ανάλογα με τον συντελεστή που θα βάλουμε, το νέο περίγραμμα που θα προκύψει θα αποτελείται από λιγότερα σημεία και συνεπώς θα είναι πιο γρήγορη η ταχύτητα εκτέλεσης. Προσοχή όμως χρειάζεται ώστε όταν δοθεί έναν μεγάλο νούμερο π.χ της τάξης του 10% τότε θα έχουμε σημαντικά λιγότερη ακρίβεια αλλά μεγαλύτερη ταχύτητα στον υπολογισμό. Στη συγκεκριμένη περίπτωση χρησιμοποιήσαμε έναν πολύ μικρό συντελεστή (0.0002) γιατί προτιμήσαμε να εστιάσουμε στην ακρίβεια κυρίως παρά στην ταχύτητα εκτέλεσης. Όλοι οι επόμενοι υπολογισμοί γίνονται με την εκτιμώμενη αυτή επιφάνεια (approximated contour).

Για την εμφάνιση του περιγράμματος της επιφάνειας που προκύπτει από τις ανωτέρω συναρτήσεις χρησιμοποιούμε την εξής συνάρτηση:

cv2.drawContours(frame, [approx_contour], 0, (0,255,0), 1).

Τα ορίσματα που δέχεται είναι η εικόνα που βλέπει η κάμερα, η contour επιφάνεια, δείκτης για το πόσα σημεία θα σχεδιαστούνε, χρώμα του περιγράμματος και το πάχος της γραμμής. Στη συνέχεια, θα βρούμε το κέντρο του σχήματος του χεριού. Αυτό γίνεται με τη χρήση της συνάρτησης cv2.moments(approx_contour). Από το documentation της OpenCV

(https://docs.opencv.org/master/dd/d49/tutorial_py_contour_features.html) βλέπουμε ότι το κέντρο ενός ακανόνιστου σχήματος (blob) έχει συντεταγμένες :

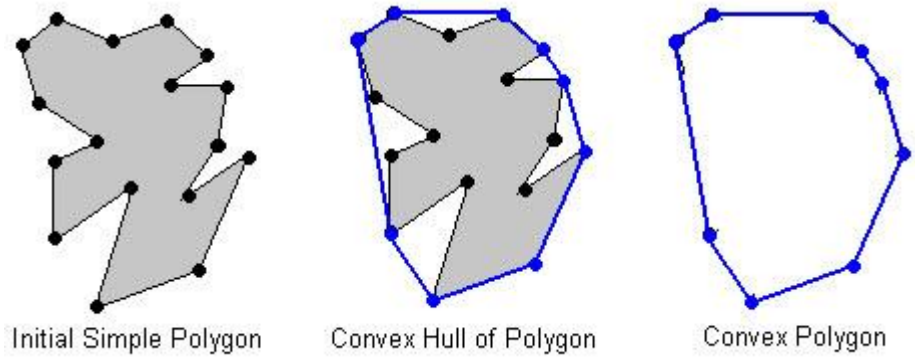
centerX = int(M['m10']/M['m00'])

centerY = int(M['m01']/M['m00'])

Το κέντρο του χεριού το εμφανίζουμε ως κύκλο με την εντολή cv2.circle() με ορίσματα την εικόνα (frame), τις συν/νες του κέντρου του κύκλου, την ακτίνα και το χρώμα.

Μερικές ιδιαίτερα χρήσιμες συναρτήσεις που μας δίνει η OpenCV είναι η **cv2.convexHull() και η cv2.convexityDefects**() [B5]. Η πρώτη λειτουργεί ως εξής:

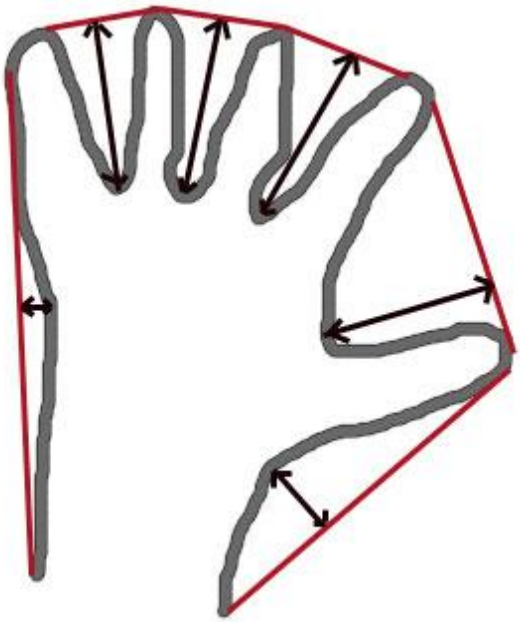
Η convexHull επιστρέφει σημεία της επιφάνειας που μελετάμε (approx_contour) στα οποία έχουν εξομαλυνθεί οι εσοχές, δηλαδή μπορούμε να το φανταστούμε ως ένα λάστιχο που πιάνεται στα ακραία σημεία του εκάστοτε σχήματος-επιφάνειας όπως στο παρακάτω σχήμα.



Εικόνα 42: Απεικόνιση της συνάρτησης Convex Hull της OpenCV

(Πηγή: https://gurus.pyimagesearch.com/lesson-sample-advanced-contour-properties/)

Με τη χρήση του function της OpenCV convexityDefects, βρίσκουμε τις εσοχές που δημιουργούνται μεταξύ της επιφάνειας που μελετάμε (approx_contour) και του convexHull όπως βλέπουμε και στο παρακάτω σχήμα.



Εικόνα 43: Απεικόνιση των συνάρτησης convexity Defects της OpenCV

(Πηγή: https://gurus.pyimagesearch.com/lesson-sample-advanced-contour-properties/)

Με κόκκινο είναι το σχήμα του convexHull και με μαύρο βέλος σημειώνονται τα σημεία που εντοπίστηκαν ως convexity Defect (εσοχές) από την αντίστοιχη συνάρτηση. Συνεπώς, με τη χρήση αυτών των 2 συναρτήσεων μπορούμε να μετρήσουμε πόσα δάχτυλα (όχι ποια όμως σε αυτό το στάδιο) εμφανίζονται αφού περιορίσουμε τις εσοχές αυτές με κάποιες συνθήκες.

Η συνάρτηση που φτιάξαμε με όνομα contours_calc δέχεται ορίσματα το εκάστοτε frame της κάμερας, τη φωτεινότητα των Pixels εντός του παραθύρου roi_difference, καθώς και την ασπρόμαυρη εικόνα (difference) του χεριού μας που προκύπτει από την απόλυτη αφαίρεση της αποθηκευμένης εικόνας χωρίς το χέρι μας και του εκάστοτε frame της κάμερας.

```
def contours_calc(frame, intensity, difference) :

        global text_pos, font, fontScale, text_color, text_thickness, text_pos7

        (contours, hierarchy) = cv2.findContours(difference, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

       #ελεγχος θερμοκρασιας cpu + μηνυμα

        cpu_temp = (os.popen("vcgencmd measure_temp").readline())
```

```
        #η temp ειναι string 11 θεσεων που γραφει temp=62.8'C

        #μετατροπη σε list

         cpu_temp = float(cpu_temp[5:9])

         if cpu_temp >= 80.5 :

              cv2.putText(frame, 'high cpu temp' , text_pos7, font, fontScale2, text_color, text_thickness, cv2.LINE_AA)  #εμφανιση μηνυματος στην lcd

         elif cpu_temp>=85 :

               sys.exit(“ Cpu overheating! ”)

          if (len(contours)== 0):  #δεν υπαρχει χερι εντος του frame

               print("no contours detected")

               cv2.putText(frame, 'no contours detected', text_pos, font, fontScale, text_color, text_thickness, cv2.LINE_AA)

               numb_fingers = 0

         elif (intensity[0] >= 25) and (intensity[0]<= 170):   #υπαρχει χερι στο παραθυρο

           # οριζουμε ότι εάν εμφανιστει και άλλη contour επιφανεια, η μεγαλυτερη θα
#             είναι το χερι

               largest_contour = max(contours, key = cv2.contourArea)   #αναλογα με το key #δηλ την επιφανεια του καθε contour, επεστρεψε την μεγαλυτερη επιφανεια

               cv2.putText(frame, 'intensity' +str(intensity), text_pos, font, fontScale, text_color, text_thickness, cv2.LINE_AA)

               epsilon = 0.0002*cv2.arcLength(largest_contour, True)      #οσο πιο μικρο #πλησιαζει το αποτελεσμα του drawcontours δηλ καμπυλωτο με αρκετα σημεια

               approx_contour = cv2.approxPolyDP(largest_contour, epsilon, True) #μειωνουμε τα contour σημεια
```

```
                    #εμφανιζουμε την επιφανεια του aprox contour στην lcd
                    cv2.drawContours(frame, [approx_contour], 0, (0,255,0), 1)
                   # βρισκουμε τις συν/νες του κεντρου του χεριου
                    M = cv2.moments(approx_contour)
                    if M["m00"] != 0:
                            centerX = int(M["m10"] / M["m00"])   #ΕΠΙΣΤΡΕφΕΙ ΣυΝ/ΝΕς
                            centerY = int(M["m01"] / M["m00"])
                    else:
                            centerX, centerY = 0, 0
                   # tuple με συν/νες κεντρου χεριου
                   Center =(centerX, centerY)
                    print('hand center is' +str(Center))
                    cv2.circle(frame, (centerX, centerY), 5, (255, 255, 0), 1)  #εμφανιζουμε κυκλο στο κεντρο χεριου
                    hull = cv2.convexHull(approx_contour, returnPoints=False)
                   #υπολογιζουμε το ανωτερο σημείο στο χερι που βλεπει η καμερα
                   Top = tuple(approx_contour[approx_contour[:, :, 1].argmin()][0])
                   fingers_calc(Top, C, hull, approx_contour, frame)     #καλει εσωτερικα την #συναρτηση fingers calc
  return()
```

Η Top μεταβλητή έχει τις συντεταγμένες του σημείου που βρίσκεται υψηλότερα στην contour επιφάνεια που μετράμε δηλαδή το χέρι μας. Αυτή έπειτα χρησιμοποιείται για να υπολογίσουμε την ευκλείδια απόσταση του υψηλότερου σημείου της contour επιφάνειας με το κέντρο του χεριού, μέσω της συνάρτησης distance.euclidean αφού εισάγουμε τη

βιβλιοθήκη **from scipy.spatial import distance**. Όταν η απόσταση αυτή είναι μικρότερη από 190 τότε σημαίνει ότι δεν υπάρχει κανένα υψωμένο δάχτυλο μπροστά από την κάμερα, δηλαδή έχουμε κλειστή παλάμη. Οι υπολογισμοί αυτοί ήταν αποτελεσματικοί καθώς δημιουργήσαμε ένα bounding box (πλαίσιο) όπου ο χρήστης πρέπει εντός του, να τοποθετήσει το χέρι του, περιορίζοντας έτσι τον τρόπο και την απόσταση από την κάμερα που θα πρέπει να το τοποθετήσει. Έπειτα εμφανίζουμε το αντίστοιχο μήνυμα στην οθόνη μας μέσω της εντολής **cv2.putText()** και στέλνουμε σειριακά μέσω της εντολής **serial.write** τον χαρακτήρα 'A' όπου θα δηλώνει την κλειστή παλάμη. Σε διαφορετική περίπτωση έχουμε τουλάχιστον ένα δάχτυλο στην κάμερα.

Στη συνέχεια καλούμε τη συνάρτηση (function) fingers_calc():

```
def  fingers_calc(Top, Center, hull, approx_contour, frame) :

        global   text_pos2, font, fontScale, text_color, text_thickness

        distance_from_center = distance.euclidean(Top, C)

        if (distance_from_center <  190) :  #εφ οσον το χερι βρισκεται ολοκληρο μεσα στο
#boundingbox  περιπου 150-190 τιμη η κλειστη παλαμη

                numb_fingers = 0     #δεν εχει κανει εκταση καποιο δαχτυλο

                cv2.putText(frame, 'closed palm' , text_pos3, font, fontScale, text_color,
text_thickness, cv2.LINE_AA)

                 serial.write(b'A')



            else :

                if  len(hull) > 2:

                    defects = cv2.convexityDefects(approx_contour, hull)

                    fingers_array1 = [ ]   #περιεχει ολα τα δαχτυλα -1 (του πρωτου δηλ thumb
#κοκ)
```

```
                    fingers_array2 = [ ]      #περιεχει μονο 1 δαχτυλο.thumb ή το αμεσως
#επομενο από δεξια προς τα αριστερά

                    fingers_array = [ ]    #πινακας με τις θεσεις  ολων των δαχτυλων

                    convex_defects_array = []

                    fingers_angle = []      #list για γωνιες μεταξυ δαχτυλων

                    for i in range(defects.shape[0]):

                        s, e, f, d = defects[i, 0]

                        start = tuple(approx_contour[s, 0])

                        end = tuple(approx_contour[e, 0])

                        far = tuple(approximated_contour[f, 0])

                        a = math.sqrt((end[0] - start[0])**2 + (end[1] - start[1])**2)

                        b = math.sqrt((far[0] - start[0])**2 + (far[1] - start[1])**2)

                        c = math.sqrt((end[0] - far[0])**2 + (end[1] - far[1])**2)

                         angle = math.acos((b**2 + c**2 - a**2) / (2*b*c))

                         fingers_angle.append(angle)

                           #εάν η γωνια είναι < από 90 μοιρες τοτε είναι εσοχη

                            if angle<= math.pi/2:

                                cv2.circle(frame,end,5,[255,0,255],-1)   #δεν εμφανιζει τον
#αντιχειρα

                                 cv2.circle(frame,far,5,[255,0,255],-1)    #εμφανιζει  όλα τα
#convex defects



                                 fingers_array1.append(list(end))
```

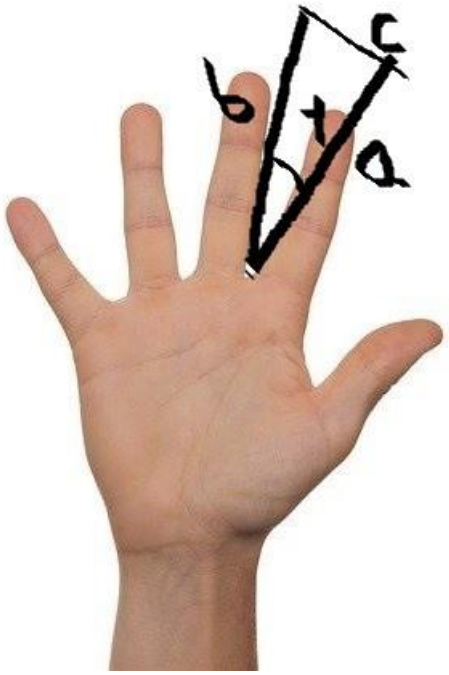
```
                                fingers_array2.append(list(start))

                                convex_defects_array.append(far)   #περιεχει τα convexity
#defects



                try:

                    numb_fingers = min(max((len(convex_defects_array)+1), 0), 5)
#περιορισμος τιμων απο 1 εως 5

                    print("fingers are :" +str(numb_fingers))

                    if (numb_fingers == 1) :

                        ser.write(b'B')

                        cv2.putText(frame, 'index only' , text_pos3, font, fontScale,
text_color, text_thickness, cv2.LINE_AA)

                    first_finger = np.array(fingers_array2[0])      #προσθετουμε το 1ο
#στοιχειο του πινακα start στον πινακα end.

                    cv2.circle(frame, tuple(first_finger) ,5,[255,0,255],-1)      #και το
#εμφανιζουμε

                    fingers_array = np.vstack ((first_finger, fingers_array1) ) #Περιεχει
#πλεον ολες τις θεσεις των δαχτυλων. Πρωτο στο List το thumb που εχει μεγαλυτερο χ απο
#τα αλλα

                     distance_2fingers = distance.euclidean(fingers_array[0],
fingers_array[1])  #αποσταση μεταξυ πρωτου και δευτερου δαχτ. απο δεξια προς αριστερά

                       print('distance between 1st and 2nd finger is ' +str(distance_2fingers))

                    #καλουμε το function fingers serial για υπολογισμο και αποστολη
#σειριακα των δεδομενων
```

```
                    fingers_serial(numb_fingers, fingers_array, distance_2fingers, Center,
fingers_angle)
            except IndexError as error:
                print("row 2[0] error")
                pass
            cv2.putText(frame, 'fingers are: ' + str(numb_fingers), text_pos2, font, fontScale,
text_color, text_thickness, cv2.LINE_AA)
        return()
```

Αφού υπολογιστούν οι εσοχές μέσω της **convexity Defects μας επιστρέφει 4 πίνακες** όπου κάθε γραμμή περιέχει τα εξής:

- Η 1η γραμμή περιέχει τις συντεταγμένες x,y όλων των σημείων των εσοχών που σχηματίζονται εκτός από το τελευταίο (s μεταβλητή).
- Η 2η γραμμή περιέχει τις συντεταγμένες x,y όλων των σημείων των εσοχών που σχηματίζονται εκτός από το πρώτο (e μεταβλητή).)
- Η 3η γραμμή περιέχει όλες τις συντεταγμένες x,y όλων των σημείων των εσοχών που σχηματίζονται (f μεταβλητή).
- Τέλος, η 4η γραμμή περιέχει όλες τις αποστάσεις όλων των σημείων των εσοχών που σχηματίζονται (d μεταβλητή). Για να βρούμε εάν αυτή η εσοχή είναι σημείο όπου έχει γίνει έκταση του δαχτύλου χρησιμοποιούμε το **θεώρημα των ημιτόνων** [Δ6 σελ.45-46] όπου για να βρούμε τη γωνία που σχηματίζεται μεταξύ των δαχτύλων ισχύει ο παρακάτω μαθηματικός τύπος:

$$\gamma = \cos^{-1}\left(\frac{a^2+b^2 - c^2}{2\,a\,b}\right)$$



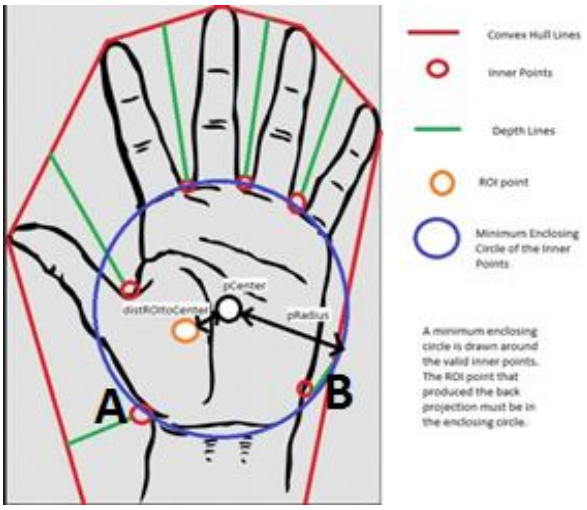
Εικόνα 44: Υπολογισμός γωνίας μεταξύ δυο δαχτύλων.

(Πηγή: https://medium.com/analytics-vidhya/hand-detection-and-finger-counting-using-opencv-python-5b594704eb08)

Η γωνία βρίσκεται από τους παρακάτω τύπους [Δ10]:

```
a = math.sqrt((end[0] - start[0])**2 + (end[1] - start[1])**2)

b = math.sqrt((far[0] - start[0])**2 + (far[1] - start[1])**2)

c = math.sqrt((end[0] - far[0])**2 + (end[1] - far[1])**2)

        angle = math.acos((b**2 + c**2 - a**2) / (2*b*c))
```

Ένα πρόβλημα που αντιμετωπίσαμε είναι το παρακάτω. Η convexity defects εντοπίζει πολλές φορές και τα σημεία Α και Β τα οποία δημιουργούνε εσοχές αλλά δεν ανήκουν στις εσοχές που περικλείονται εντός των δαχτύλων, συνεπώς πρέπει να αφαιρεθούνε. Αυτό γίνεται αφού βρούμε την μεταβλητή angle για κάθε δάχτυλο η οποία θα πρέπει να έχει τιμή μικρότερη από 90 μοίρες.



Εικόνα 45: Επιπλέον σημεία από τη χρήση της συνάρτησης convexity Defects

(Πηγή: http://eprints.utar.edu.my/3055/1/fyp_IA_2018_CYX_-_1403243.pdf σελ.9)

Εφ' όσον η γωνία είναι μικρότερη από 90 μοίρες τότε εμφανίζουμε κύκλους στα αρχικά και τελικά σημεία των εσοχών.

Εάν δεν εντοπιστούνε σωστά τα ανιχνευμένα σημεία μπορεί να διακοπεί η εκτέλεση του προγράμματος και να εμφανιστεί σφάλμα στο command window, συνεπώς δημιουργούμε μια συνθήκη για **error handling try-except** όπου αρχικά περιορίζουμε τον αριθμό των μετρούμενων δαχτύλων από 1 έως 5, στέλνουμε σειριακά **ser.write(b'B')** στον Arduino, και εμφανίζουμε το αντίστοιχο μήνυμα στην οθόνη μας.

Για να πάρουμε στη λίστα fingers_array όλες τις συντεταγμένες των εσοχών θα πάρουμε το πρώτο στοιχείο του πίνακα με τις start συντεταγμένες και θα το προσθέσουμε στη λίστα με τις end συντεταγμένες και θα τις αποθηκεύσουμε στο fingers_array list. Στη συνέχεια, υπολογίζουμε και την ευκλείδια απόσταση μεταξύ του πρώτου και δεύτερου δαχτύλου καθώς θα χρησιμοποιηθεί στους επόμενους υπολογισμούς. **Πρέπει να σημειωθεί ότι όλοι οι ανωτέρω υπολογισμοί έχουν γίνει για το δεξί χέρι, συνεπώς τοποθετώντας το αριστερό χέρι στην κάμερα θα δώσει λάθος αποτελέσματα στην έξοδο (ρομποτικό χέρι).**

Τέλος, καλούμε εσωτερικά ένα ακόμη function με ονομασία fingers_serial όπου εντός του υπολογίζουμε ποια δάχτυλα είναι αυτά και στέλνουμε τα δεδομένα αυτά στον arduino.

Η συνάρτηση αυτή παίρνει τα εξής ορίσματα:

```
def  fingers_serial(numb_fingers, fingers_array, distance_2fingers, Center, fingers_angle)
```

Εντός της, έχουμε όλες τις περιπτώσεις όπου η numb_fingers μπορεί να πάρει τις τιμές 0 έως 5 (κλειστή παλάμη έως ανοιχτή παλάμη). Όταν η numb_fingers πάρει τιμές 2, 3 ή 4 έχουμε τις εξής υποπεριπτώσεις:

Αριθμός δαχτύλων	**Υποπεριπτώσεις**
2	Δείκτης, μικρός (Index, Pinky)
2	Δείκτης, μεσαίος (Index, Middle)
2	Αντίχειρας, δείκτης (Thumb, Index)
3	Αντίχειρας, δείκτης, μεσαίος (Thumb, Index, Middle)
3	Αντίχειρας, δείκτης, μικρός (Thumb, Index, Pinky)
3	Δείκτης, μεσαίος, παράμεσος (Index, Middle, Ring)
3	Μεσαίος, παράμεσος, μικρός (Middle, Ring, Pinky)
4	Δείκτης, μεσαίος, παράμεσος, μικρός (Index, Middle, Ring, Pinky)
4	Αντίχειρας, δείκτης, μεσαίος, μικρός (Thumb, Index, Middle, Pinky)
4	Αντίχειρας, μεσαίος, παράμεσος, μικρός (Thumb, Middle, Ring, Pinky)

Πίνακας 10: Πίνακας υποπεριπτώσεων εκτεινόμενων δαχτύλων

4.3.4 Εξέταση υποπεριπτώσεων

<u>Για 2 δάχτυλα:</u>

Όταν ανιχνευθούνε 2 δάχτυλα, η list fingers_array μας επιστρέφει τις συντεταγμένες οποιονδήποτε 2 δαχτύλων εντοπίσει. Για να καταλήξουμε σε ποια από τις 3 ανωτέρω περιπτώσεις ανήκει αρχικά ελέγχουμε τις συντεταγμένες του δεύτερου δαχτύλου και εάν είναι εντός κάποιων ορίων ως προς τις συντεταγμένες του κέντρου της παλάμης, τότε πρόκειται για την περίπτωση ***Δείκτης, μικρός (Index, Pinky).*** Σε διαφορετική περίπτωση, εάν η διανυσματική απόσταση του $1^{ου}$ δαχτύλου και του $2^{ου}$ δαχτύλου είναι εντός κάποιων ορίων καθώς και η γωνία που σχηματίζουν μεταξύ τους είναι από 10-45 μοίρες, τότε πρόκειται για την περίπτωση ***δείκτης, μεσαίος (Index, Middle),*** αλλιώς πρόκειται για την περίπτωση ***αντίχειρας, δείκτης (Thumb, Index)*** .

Για 3 δάχτυλα:

Από τις 4 περιπτώσεις που πρέπει να ελέγξουμε, οι 2 από αυτές περιέχουν τα πρώτα δάχτυλα δηλαδή τον αντίχειρα και τον δείκτη. Στις υπόλοιπες 2 περιπτώσεις δεν συμπεριλαμβάνονται τα ανωτέρω δάχτυλα. Αυτό το βρίσκουμε από την εξής συνθήκη: Εάν η τεταγμένη ψ του $1^{ου}$ δαχτύλου είναι περίπου από 80-140% της τεταγμένης ψ του κέντρου της παλάμης και η γωνία που σχηματίζουν το $1^{ο}$ με το $2^{ο}$ δάχτυλο είναι από 45 έως 85 μοίρες τότε έχει ανιχνευτεί ο αντίχειρας (Α). Έπειτα αφού ικανοποιηθεί η συνθήκη (Α) ελέγχουμε την τετμημένη του $3^{ου}$ δαχτύλου από την τετμημένη του κέντρου της παλάμης και εάν είναι εντός κάποιων ορίων τότε το $3^{ο}$ δάχτυλο είναι ο μεσαίος, ενώ σε διαφορετική περίπτωση είναι ο μικρός. Σε περίπτωση που η συνθήκη Α δεν ισχύει, ελέγχουμε την τετμημένη του $3^{ου}$ δαχτύλου που εντοπίστηκε, με την τετμημένη του κέντρου της παλάμης και ανάλογα επιλέγετε η περίπτωση ***Μεσαίος, παράμεσος, μικρός (Middle, Ring, Pinky)*** ή ***Δείκτης, μεσαίος, παράμεσος (Index, Middle, Ring)***.

Για 4 δάχτυλα:

Σε αυτή την περίπτωση, αρχικά βρίσκουμε τον αντίχειρα με την συνθήκη ότι εάν η γωνία που σχηματίζει το $1^{ο}$ και $2^{ο}$ δάχτυλο που ανιχνεύουμε είναι από 45 έως 85 μοίρες οπότε και υπάρχει ο αντίχειρας (Β). Αυτό με τη σειρά του σημαίνει ότι έχουμε τις περιπτώσεις ***Αντίχειρας, δείκτης, μεσαίος, μικρός (Thumb, Index, Middle, Pinky)*** και ***Αντίχειρας, μεσαίος, παράμεσος, μικρός (Thumb, Middle, Ring, Pinky).*** Εάν η $3^{η}$ γωνία (μεταξύ $3^{ου}$ και $4^{ου}$ δαχτύλου) είναι μικρότερη από περίπου 20 μοίρες καθώς επίσης και η ευκλείδια απόσταση μεταξύ του $3^{ου}$ και $4^{ου}$ δαχτύλου είναι εντός κάποιων ορίων τότε έχουμε την περίπτωση ***Αντίχειρας, μεσαίος, παράμεσος, μικρός (Thumb, Middle, Ring, Pinky)*** αλλιώς έχουμε την περίπτωση ***Αντίχειρας, δείκτης, μεσαίος, μικρός (Thumb, Index, Middle, Pinky).*** Όταν δεν ισχύει η συνθήκη Β, τότε έχουμε την περίπτωση ***Δείκτης, μεσαίος, παράμεσος, μικρός (Index, Middle, Ring, Pinky)*** .

Παρακάτω ακολουθεί η συνάρτηση που δημιουργήσαμε για τον υπολογισμό της εκάστοτε περίπτωσης από τον πίνακα 9 (σελ. 86) και αντίστοιχα στέλνει σειριακά τον κατάλληλο χαρακτήρα (char) στον Arduino Uno της βάσης για την κίνηση του ρομποτικού χεριού.

```
def fingers_serial(numb_fingers, fingers_array, distance_2fingers, C, fingers_angle): #C -> συν/νες κέντρου παλάμης
    global text_pos3, font, fontScale, text_color, text_thickness, text_pos4, text_pos5
    if (numb_fingers  == 0):
            print('closed palm')
            cv2.putText(frame, 'closed palm' , text_pos3, font, fontScale, text_color, text_thickness, cv2.LINE_AA)
             ser.write(b'A')
    elif (numb_fingers  == 1):
             print('index only')
             cv2.putText(frame, 'index' , text_pos3, font, fontScale, text_color, text_thickness, cv2.LINE_AA)
             ser.write(b'B')
    #για 2 δάχτυλα περιπτώσεις
    elif (numb_fingers  == 2):
            print('2 fingers')
            if (fingers_array[1][0] >= 0.25 * C[0]) and (fingers_array[1][0] <= 0.70 * C[0]) and fingers_array[0][0] > C[0]   : #συν/μενη χ του δευτερου δαχτυλου και συν/μενη χ του πρωτου δαχτυλου σε σχέση με αυτή του κέντρου της παλάμης
                    print('2 fingers are index and pinky')
                    cv2.putText(frame, 'index and pinky' , text_pos3, font, fontScale, text_color, text_thickness, cv2.LINE_AA)
                    ser.write(b'C')
            else :
```

```
                if (distance_2fingers >= 60) and (distance_2fingers <= 165) and ((fingers_angle[0] >= 0.0555 *math.pi) and (fingers_angle[0] <0.25 * math.pi)):  #αποσταση 1ου-#2ου δαχτυλου και γωνία από 10 έως 45 μοίρες

                    print("2 fingers are index and middle ")

                    cv2.putText(frame, 'index and middle' , text_pos3, font, fontScale, text_color, text_thickness, cv2.LINE_AA)

                    ser.write(b'D')

                elif (distance_2fingers >= 165) and ((fingers angle[0] >= 0.25 * math.pi) and (fingers_angle[0] <=0.472 *math.pi)) :

                    print('2 fingers are thumb and index')

                    cv2.putText(frame, 'thumb and index' , text_pos3, font, fontScale, text_color, text_thickness, cv2.LINE_AA)

                    ser.write(b'E')

  #για 3 δάχτυλα περιπτώσεις

   elif(numb_fingers == 3):

            print('3 fingers')

            # print('3rd finger x is : ' +str(fingers_array[2][0]))

            if  (fingers_array[0][1] <= C[1]* 0.8) and fingers_array[0][1] >= C[1]*1.4):

            # αποσταση ψ συν/νης 1ου δαχτυλου απο το κεντρο του χεριου

                if (fingers_array[2][0] >= (C[0] - (290 / C[0] * 100)) and (fingers_array[2][0] <= (C[0] + (290 / C[0] * 100)))):

                    cv2.putText(frame, 'thumb, index, middle' , text_pos4, font, fontScale, text_color, text_thickness, cv2.LINE_AA)

                    print('3 fingers are thumb, index, middle')
```

```
                    ser.write(b'F')
                else :
                    cv2.putText(frame, 'thumb, index, pinky' , text_pos3, font, fontScale, text_color, text_thickness, cv2.LINE_AA)
                    print('3 fingers are thumb, index, pinky')
                    ser.write(b'G')
            else :
                if (fingers_array[2][0] >= 0.62 * C[0]) and (fingers_array[2][0] <= 0.95 * C[0]):
                    print("3 fingers are index, middle, ring ")
                    cv2.putText(frame, 'index, middle, ring' , text_pos3, font, fontScale, text_color, text_thickness, cv2.LINE_AA)
                    ser.write(b'H')
                elif (fingers_array[2][0] >= 0.31 * C[0]) and (fingers_array[2][0] <= 0.605 * C[0]):
                    print('3 fingers are middle, ring, pinky')
                    cv2.putText(frame, 'middle, ring, pinky' , text_pos3, font, fontScale, text_color, text_thickness, cv2.LINE_AA)
                    ser.write(b'I')

                else :
                    print('unknown case')
                    cv2.putText(frame, 'unknown case' , text_pos3, font, fontScale, text_color, text_thickness, cv2.LINE_AA)
```

```
   elif(numb_fingers  == 4):
                print('4 fingers')
                distance_3fingers = distance.euclidean(fingers_array[2], fingers_array[3])
                print('distance between 3rd and fourth finger is ' +str(distance_3fingers))
                 if ((fingers_angle[0] >= 0.25 * math.pi) and (fingers_angle[0] <=0.472 *math.pi)):  #εαν η γωνία από 45 έως 85 μοίρες τότε ο αντίχειρας έχει ανιχνευθεί
                            if (fingers_angle[2] <=0.1 *math.pi) and (distance_3fingers <= 130):
                                    print("4 fingers are thumb, middle, ring,  pinky ")
                                    cv2.putText(frame, ‘thumb, middle, ring,   pinky ‘ , text_pos5, font, fontScale, text_color, text_thickness, cv2.LINE_AA)
                                    ser.write(b'L')
                            else:
                                     print("4 fingers are thumb, index, middle,  pinky ")
                                    cv2.putText(frame, thumb, index, middle,   pinky ' , text_pos5, font, fontScale, text_color, text_thickness, cv2.LINE_AA)
                                    ser.write(b'K')
                    else:
                            print("4 fingers are index, middle, ring, pinky ")
                            cv2.putText(frame, index, middle, ring, pinky' , text_pos5, font, fontScale, text_color, text_thickness, cv2.LINE_AA)
                                   ser.write(b'J')
```

```
    elif (numb_fingers  == 5):
                print('5 fingers')
                cv2.putText(frame, 'open palm' , text_pos3, font, fontScale, text_color, text_thickness, cv2.LINE_AA)
                 ser.write(b'M')
    else:
                print("not known condition")
    return()
```

4.3.5 Arduino Uno - receiver λειτουργία

Αφού γίνει μέσω της OpenCV η επεξεργασία του εκάστοτε frame και σταλθούν σειριακά τα δεδομένα από το raspberry στο Arduino Uno, αυτός με τη σειρά του διαβάζει αυτά τα byte που στάλθηκαν και στέλνει στο PCA 9865 τα κατάλληλα σήματα για την κίνηση των σερβοκινητήρων.

Δημιουργούμε έναν δισδιάστατο πίνακα που περιέχει τιμές 150 ή 500 για κάθε μια από τις 13 περιπτώσεις που αναγνωρίζει το Raspberry.

Για όλα τα δάχτυλα το 500 αντιστοιχεί σε κλειστή θέση. Παρομοίως για ανοιχτή θέση, όλα τα δάχτυλα δέχονται την τιμή 150 στη συνάρτηση setPWM του αντικειμένου Adafruit_PWMServoDriver (PCA9685).

```
//Για κάθε περιπτωση (γραμμα) αντιστοιχει πινακας παλμων 5 θεσεων απο το raspberry που στελνει μεσω σειριακης usb
//T = Thumb,  I = Index,  M = Middle,  R = Ring,  P = Pinky
const int PulsesArray[13][5] ={  {500, 500, 500, 500, 500},  //char A = 0 fingers
          {500, 150, 500, 500, 500},     //char B = 1 finger - I
           {500, 150, 500, 500, 150},     //char C = 2 fingers – I,P
          {500, 150, 150, 500, 500},     //char D = 2 fingers –I,M
```

```
          {150, 150, 500, 500, 500},     //char E = 2 fingers –T,I
          {150, 150, 150, 500, 500},     //char F = 3 fingers –T,I,M
          {150, 150, 500, 500, 150},     //char G = 3 fingers –T,I,P
          {500, 150, 150, 150, 500},     //char H = 3 fingers –I,M,R
          {500, 500, 150, 150, 150},     //char I = 3 fingers –M,R,P
          {500, 150, 150, 150, 150},     //char J = 4 fingers –I,M,R,P
          {150, 150, 150, 500, 150},     //char K = 4 fingers – T, I, M, P
          {150, 500, 150, 150, 150},     //char L = 4 fingers –T,M,R,P
          {150, 150, 150,150, 150}   };  //char  M = 5 fingers
```

Έπειτα στη void setup αρχικοποιούμε τα Pins 7 και 8 όπου όταν ενεργοποιηθεί το πρώτο ο αλγόριθμος εκτελεί το function για τον έλεγχο μέσω κάμερας και Raspberry, ενώ όταν ενεργοποιηθεί το δεύτερο, εκτελεί το function για τον έλεγχο μέσω του γαντιού. Αντίστοιχα στη void loop έχουμε:

```
Void loop() {
if (digitalRead(CameraMode) == HIGH) {
// raspberry camera mode
          camera_mode();
 }
else if (digitalRead(GloveMode) == HIGH) {
          glove_mode();
  }
else {
   //κλειστη θεση χεριου
```

```
            PwmSignals(PulsesArray, 0);  //1η σειρα του 2d πινακα Pulses Array

        }

}
```

Επιπροσθέτως, οι ανωτέρω τιμές 150 και 500 δίνονται ως όρισμα στην εντολή pwm.setPWM(κανάλι σερβοκινητήρα, 0, τελική θέση) που δίνει κίνηση στους κινητήρες μέσω της βιβλιοθήκης <Adafruit_PWMServoDriver.h>.

Το function που φτιάξαμε για τον έλεγχο μέσω κάμερας.

```
Void camera_mode() {

if (Serial.available()> 0) {

     char  r = Serial.read();

      if (r == 'A') {        //char A = 0 fingers - closed palm

           PwmSignals(PulsesArray, 0);  //1η σειρα του 2d πινακα Pulses Array

       }

       else if (r == 'B') {   //char B = 1 fingers - index

          PwmSignals(PulsesArray, 1);  //2η σειρα του 2d πινακα Pulses Array

       }

      else if{εξέταση για κάθε μια από τις περιπτώσεις από ‘C’ έως ‘M’}

}
```

Η υπορουτίνα PwmSignals που δημιουργήσαμε, δέχεται ως ορίσματα κάθε φορά την εκάστοτε γραμμή του δισδιάστατου πίνακα PulsesArray που αφορά κάθε μια από τις 13 στο σύνολο περιπτώσεις ανιχνευμένων δαχτύλων, ενώ εντός της τρέχει την εντολή pwm.setPWM (κανάλι σερβοκινητήρα, 0, τελική θέση).

```
//---------------S U B R O U T I N E S (PROGRAM 1 - CAMERA MODE)--------------

void PwmSignals(const int pulses[13][5], byte row) {

for (int i = row; i < row+1; i++) {

        for (int j = 0; j < 5 ; j++) {

                pwm.setPWM(j*2, 0, pulses[row][j]);

        }

    }

}
```

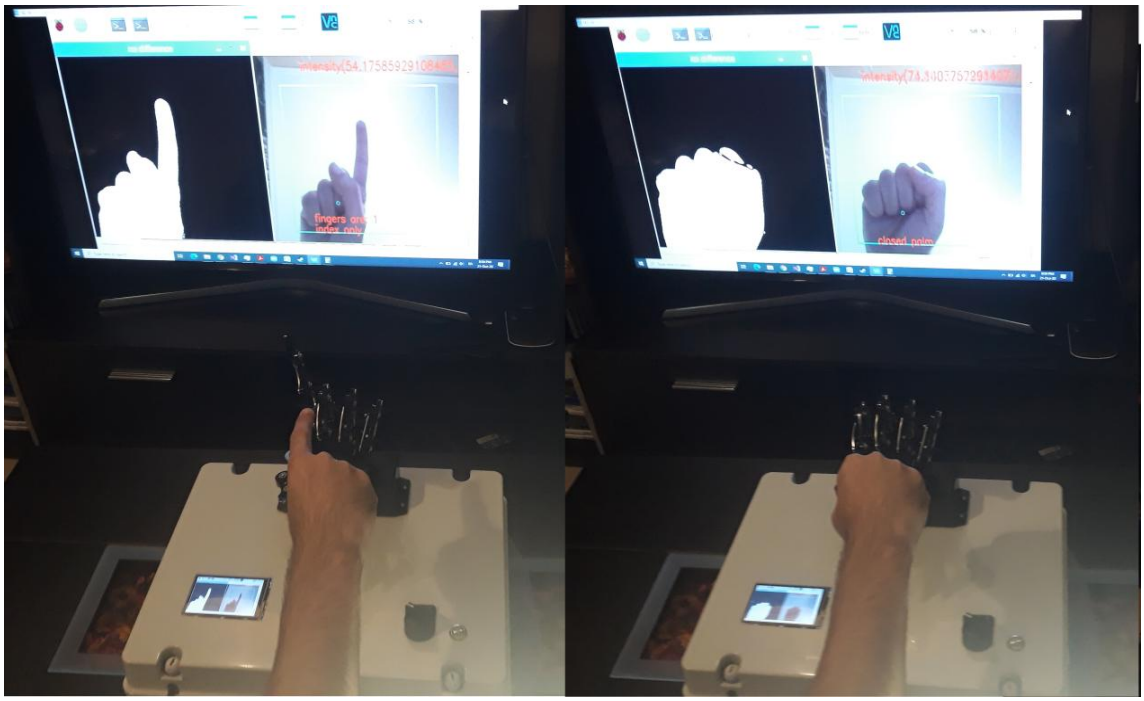
4.3.6 Εκτέλεση προγράμματος

Για την εκτέλεση του Python script που φτιάξαμε αρχικά ανοίγουμε το command line και πληκτρολογούμε την εντολή: **source ~/.profile**

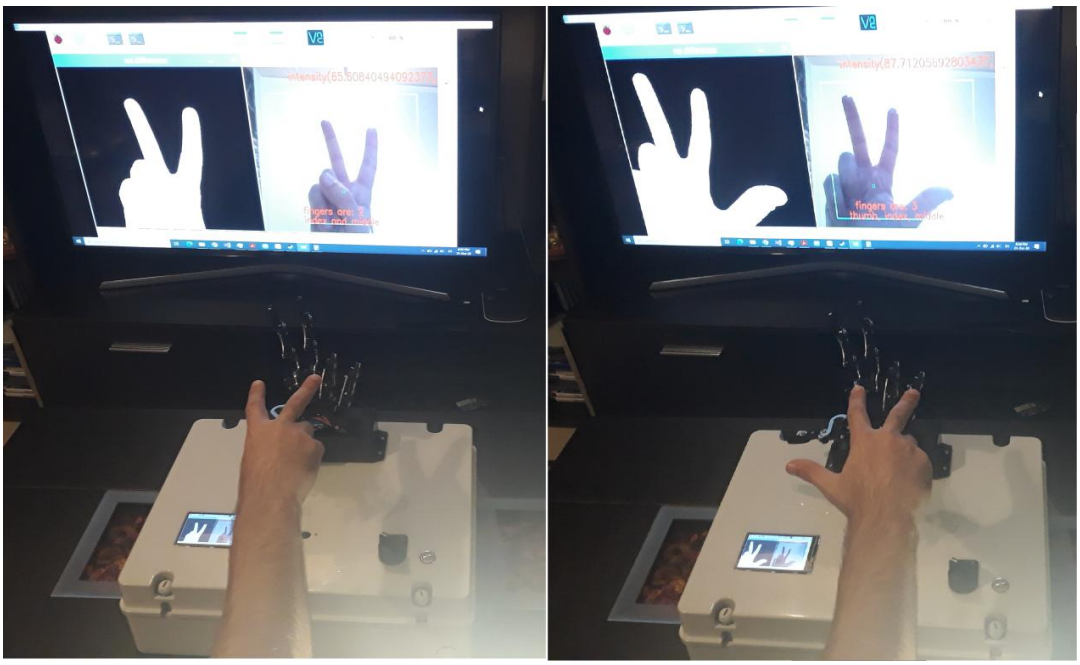
Έπειτα πληκτρολογούμε την εντολή: **workon cv (φόρτωση venv)**

Και τέλος την εντολή: **background_subtraction.py** που είναι και το όνομα του python script που δώσαμε.

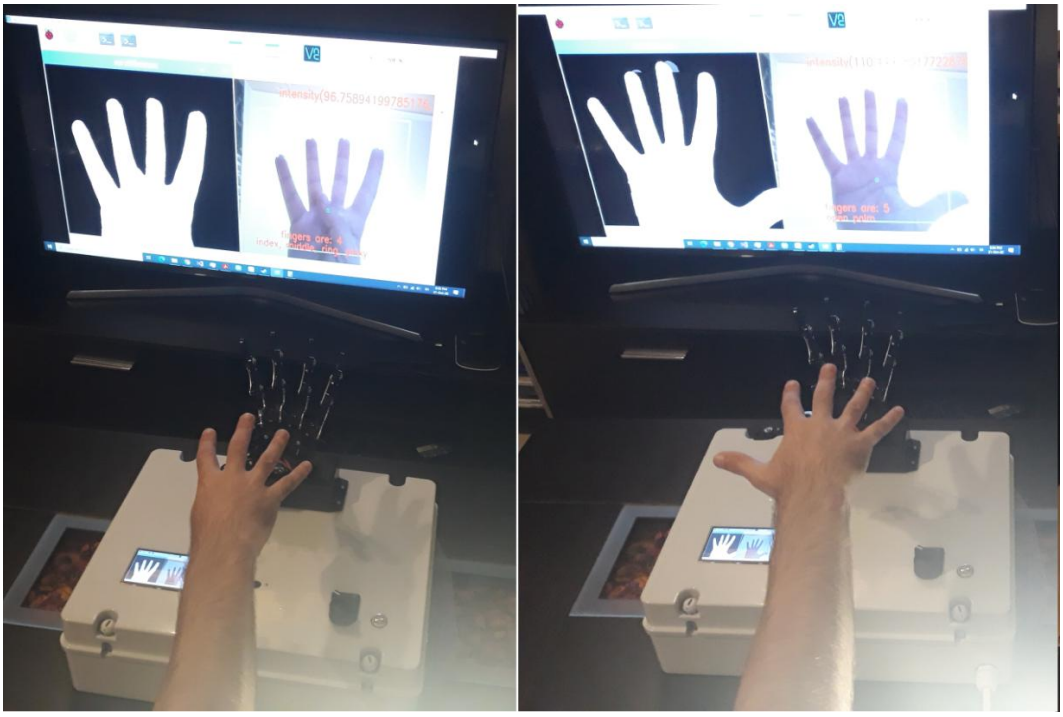
Παρακάτω παραθέτονται μερικές φωτογραφίες από την αναγνώριση των δαχτύλων του χεριού.



Εικόνα 46: Αριστερά: αναγνώριση και κίνηση δείκτη- Δεξιά: Κλειστή παλάμη.



Εικόνα 47: Αριστερά: αναγνώριση και κίνηση δείκτη και μεσαίου - Δεξιά: αναγνώριση και κίνηση αντίχειρα, δείκτη και μεσαίου.



Εικόνα 48: Αριστερά: αναγνώριση και κίνηση 4 δαχτύλων εκτός αντίχειρα- Δεξιά: αναγνώριση και κίνηση όλων των δαχτύλων.

Κεφάλαιο 5

5.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα σχολιάσουμε τα αποτελέσματα και τους περιορισμούς και για τα 2 mode ελέγχου και στο τέλος προτείνουμε βελτιώσεις πάνω σε αυτά. Θα πρέπει επίσης να τονιστεί ότι το συνολικό κόστος της εργασίας δεν είναι υψηλό αλλά είναι υπολογίσιμο (περίπου στα 550-600 €) για μια παρόμοιου τύπου κατασκευή. Οι γνώσεις που απαιτούνται για την διεκπεραίωση της εργασίας περιλαμβάνουν γνώσεις ηλεκτρονικής, ηλεκτρολογίας και προγραμματισμού (python και C++).

5.2 Συμπεράσματα- παρατηρήσεις για έλεγχο με απτικό γάντι ($1^{Ο}$ Mode ελέγχου)

Στο συγκεκριμένο Mode ελέγχου είδαμε την επίδραση του Kalman φίλτρου σε αναλογικά αισθητήρια και πως βελτιώνει σημαντικά τον τελικό έλεγχο ψαλιδίζοντας ουσιαστικά τις κορυφές και μας παρέχει μια ομαλή μεταβολή των τιμών. Το συμπέρασμα που βγάζουμε για το μονοδιάστατο Kalman φίλτρο είναι ότι είναι γρήγορο στην εκτέλεση και προσφέρει σημαντική βελτίωση για την απόρριψη των ακραίων τιμών (outliers) κατά τη δειγματοληψία από αισθητήρες.

Επίσης, οι ασύρματες επικοινωνίες εμφανίζουν ιδιαιτερότητες όπως για παράδειγμα μπορεί ανά πάσα στιγμή να υπάρξει στιγμιαία διακοπή στην επικοινωνία μεταξύ ενός master και ενός slave συνεπώς θα πρέπει να υπάρχει αλγόριθμος που να λαμβάνει σωστά τα πακέτα αυτά που μεταδίδονται χωρίς να λαμβάνονται με λάθος σειρά με συνέπεια να έχουμε τελείως λανθασμένο χειρισμό του ρομποτικού χεριού.

Όταν ληφθεί μια τιμή για κάποιο δάχτυλο, ελέγχουμε εάν αυτή η τιμή είναι μεγαλύτερη ή μικρότερη από την προηγούμενη (έκταση ή σύμπτυξη), και αυξάνουμε σταδιακά την εντολή κίνησης του σερβοκινητήρα με μικρό βήμα, μέχρι να εξισωθούνε η τωρινή τιμή με την προηγούμενη (σε κάθε κύκλο παραλαβής πακέτου τιμών), γεγονός που βοηθάει στην πιο ομαλή κίνηση των σερβοκινητήρων.

5.2.1 Παρατηρήσεις

Κατά την εκτέλεση της συγκεκριμένης λειτουργίας καταλήξαμε στις ακόλουθες παρατηρήσεις:

1) Οι αισθητήρες κάμψης τοποθετήθηκαν με Velcro στο γάντι, που έχει ως συνέπεια να μην εφάπτονται πολύ καλά στα δάχτυλα και λόγω της κίνησης να κολλάει και να ξεκολλάει ελαφρώς ανάλογα με το εάν κάμπτεται ή εκτείνεται το δάχτυλο. Αυτό επηρεάζει την αναλογική τιμή η οποία ακόμη και σε σταθερή κατάσταση παρουσιάζει υπολογίσιμες μεταβολές.
2) Λόγω της ελαστικότητας του γαντιού αρκετές φορές τα καλώδια μπορεί να χαλαρώσουν ή ακόμη και να κοπούνε.
3) Χωρίς τη χρήση του φίλτρου Kalman υπήρχε αρκετός ‘’θόρυβος’’ στα αναλογικά σήματα παρόλο που περιορίζαμε τις τιμές μέσω των built in function map και constrain. Αυτό είχε ως συνέπεια κακό και ‘’νευρικό’’ έλεγχο του ρομποτικού χεριού.
4) Η παραμετροποίηση των τιμών για το Kalman φίλτρο αγγίζει τα όρια της χρυσής τομής. Αυτό σημαίνει ότι ορίζοντας μια τιμή για το σφάλμα της μέτρησης αρκετά μεγαλύτερη από την τιμή της εκτίμησης του σφάλματος (error_measure > error_estimate) τότε το kalman gain είναι κοντά στο μηδέν τότε η έξοδος θα είναι σαν τιμή κοντά στην τιμή που κάνουμε εκτίμηση (previous estimate value). Όταν όμως το σφάλμα της μέτρησης είναι αρκετά μικρότερο από την τιμή της εκτίμησης του σφάλματος (error_measure < error_estimate) τότε το kalman gain είναι κοντά στο 1 οπότε η έξοδος θα έχει τιμή κοντά σε αυτήν που μετράται από τους αισθητήρες.
5) Ως μηχανολογική κατασκευή παρουσιάζει τζόγο που επιδρά στην ομαλή κίνηση του ρομποτικού χεριού και δεν έχουμε υψηλή ακρίβεια στην κίνηση.

5.3 Συμπεράσματα- παρατηρήσεις για έλεγχο με κάμερα και OpenCV (2^{Ο} Mode ελέγχου)

Σε πρώτη φάση θα πρέπει να τονίσουμε τη χρησιμότητα της βιβλιοθήκης OpenCV για την επίλυση αναρίθμητων προβλημάτων με μηχανική όραση ενώ παράλληλα είναι προσιτή στο μέσο χρήστη και δεν απαιτεί τη χρήση ισχυρού υπολογιστικού συστήματος.

Για την επίλυση του προβλήματος της αναγνώρισης των δαχτύλων που εμφανίζονται στην κάμερα, δημιουργήσαμε ένα πλαίσιο (bounding box) για να περιορίσουμε το χρήστη σε ποιο σημείο περίπου να τοποθετήσει το χέρι του και χωρίς μεγάλες περιστροφές ώστε να χωράει κατάλληλα σε αυτό το, για να είναι εφικτή η αναγνώριση για μεγαλύτερο πλήθος περιπτώσεων.

Η επίδραση των επιπέδων φωτός και κυρίως εάν είναι τεχνητό (από κάποιον προβολέα) αποτελεί από μόνο του ένα πρόβλημα που χρειάζεται περαιτέρω δοκιμές και έρευνες καθώς συνδέεται και με την τιμή κατωφλίωσης που θα επιλέγουμε στον αλγόριθμο εκτέλεσης.

5.3.1 Παρατηρήσεις

Κατά την εκτέλεση αυτής της λειτουργίας παρατηρήθηκαν τα εξής:

- Επιβαρύνεται η CPU του Raspberry σε υπερβολικό βαθμό μετά την λειτουργία πάνω από 5 λεπτά. Η θερμοκρασία ξεπερνάει τους 85 βαθμούς και απαιτείται ψύξη με ανεμιστήρα που στην περίπτωσή μας δεν υπάρχει λόγω έλλειψης χώρου.
- Όταν έχουμε την περίπτωση ενός δαχτύλου στην κάμερα, το να βρούμε ποιο είναι γίνεται δύσκολο στην επίλυση πρόβλημα καθώς δεν μπορούμε να κάνουμε σύγκριση και συσχέτιση με άλλα δάχτυλα, ενώ παράλληλα εάν έχουμε μεγάλες μετατοπίσεις και περιστροφές του χεριού του χρήστη το πρόβλημα αυτό γίνεται ακόμη δυσκολότερο.
- Το όριο της κατωφλίωσης επηρεάζει τα αποτελέσματα. Με μικρό όριο έχουμε καλύτερη αναγνώριση σε περιβάλλον με χαμηλότερα επίπεδα φωτισμού. Με μεγάλο όριο έχουμε καλύτερη αναγνώριση των contours και σχετικά μεγαλύτερη ακρίβεια στην αναγνώριση των δαχτύλων αλλά χρειάζεται καλύτερο φωτισμό χωρίς σκιάσεις.
- Ο αλγόριθμος λειτουργεί όταν ο χρήστης τοποθετήσει μόνο το δεξί του χέρι εντός του οπτικού πεδίου της κάμερας και όχι για το αριστερό.

5.4 Βελτιώσεις

Στην περίπτωση του **ελέγχου μέσω απτικού γαντιού** μπορούμε να:

- Τοποθετήσουμε ένα γυροσκόπιο για να βρούμε και τις κινήσεις στο χώρο (roll, pitch, yaw) και αντίστοιχα να περιστρέφει πάνω-κάτω (pitch), αριστερά-δεξιά (yaw) και περιστροφή ως προς τον καρπό (roll).
- Η καλύτερη τοποθέτηση των αισθητήρων κάμψης μπορεί να βελτιώσει την ακρίβεια των κινήσεων.
- Επειδή το γάντι είναι ελαστικό τα καλώδια (jumpers) που τοποθετήθηκαν με κόλληση (soldering) μπορούνε να κοπούνε ή και λόγω των Pin headers οι συνδέσεις δεν είναι αρκετά σφιχτές. Συνεπώς προτείνεται η τύπωση του κυκλώματος για αποφυγή των προβλημάτων αυτών.
- Χρησιμοποιήσουμε ένα ρομποτικό χέρι με καλύτερους ποιοτικά σερβοκινητήρες που θα μπορούνε να κάνουν και έλεγχο ροπής, να έχουν μεγαλύτερη ακρίβεια ενώ παράλληλα ως μηχανολογική κατασκευή οι σύνδεσμοι να παρουσιάζουν λιγότερο τζόγο (backlash).
- Βάζουμε σε **λειτουργία ύπνου (sleep mode)** το Arduino Nano και να μην στέλνει τιμές κατευθείαν το HC05 master Bluetooth όταν συνδέουμε την τροφοδοσία, παρά μόνο εάν παρατηρηθεί αλλαγή στις τιμές των αισθητήρων κάμψης, πραγματοποιώντας περαιτέρω εξοικονόμηση ενέργειας.
- Με χρήση σερβοκινητήρων ελέγχου ροπής καθώς και αισθητήρων ηλεκτρομυογραφήματος που μετράνε το σήμα των νεύρων και το ενισχύουν, μπορεί το ρομποτικό χέρι να πιάνει ένα αντικείμενο με συγκεκριμένη δύναμη ανάλογα με τη σύσταση του υλικού που πρόκειται να αγγίξει-κρατήσει.

Αντίστοιχα στην περίπτωση του **ελέγχου μέσω κάμερας** μπορούμε να κάνουμε τις εξής βελτιώσεις:

- Να αλλάξουμε τρόπο υλοποίησης και να χρησιμοποιήσουμε τις βιβλιοθήκες tensorflow για τη χρήση machine learning που είναι πιο κατάλληλα για παρόμοιου είδους προβλήματα.
- Θα μπορούσαμε να χρησιμοποιήσουμε έναν Arduino Mega που περιέχει αρκετές PWM εξόδους και θα ήταν πλέον περιττή η χρήση του PCA9685 servodriver, ενώ επίσης επειδή διαθέτει περισσότερες από 1 UART θύρες (σε αντίθεση με τον Uno ο οποίος διαθέτει μόνο μια) θα καθιστούσε περιττό και το κύκλωμα με το ρελέ που ενεργοποιεί/απενεργοποιεί τη σειριακή επικοινωνία από τα pins 0 και 1.
- Εφαρμογή αναγνώρισης συγκεκριμένων χειρονομιών από τη νοηματική για την ευκολότερη και ταχύτερη εξυπηρέτηση και εκπαίδευση κωφάλαλων ατόμων.
- Καθώς πρόκειται για αφηρημένο πρόβλημα, συνίσταται η χρήση ενός classifier μηχανικής μάθησης (machine learning) όπως για παράδειγμα η χρήση του haar cascade classifier που μας παρέχει η OpenCV.

Βιβλιογραφία

- [B1] Thomas B. Moeslund and Lau Nørgaard, "*A Brief Overview of Hand Gestures used in Wearable Human Computer Interfaces"* Archived 2011-07-19 at the Wayback Machine, Technical report: CVMT 03-02, ISSN 1601-3646, Laboratory of Computer Vision and Media Technology, Aalborg University, Denmark.
- [B2] *Hand Gesture Recognition for Contactless Device Control in Operating Rooms* Ebrahim Nasr-Esfahani, Nader Karimi, S.M. Reza Soroushmehr, M. Hossein Jafari, M. Amin Khorsandi, Shadrokh Samavi, Kayvan Najarian
- [B3] Michael, Beyeler 2015. *OpenCV with Python blueprints*. Birmingham: Packt Publishing
- [B4] M Sreelakshmi, TD Subash. 2017. *Haptic technology: A comprehensive review on its applications and future prospects* - Materials Today: Proceedings, Elsevier
- [B5] Steger, Carsten; Markus Ulrich; Christian Wiedemann (2018). *Machine Vision Algorithms and Applications* (2nd ed.)
- [B6] Inrak Choi1, Heather Culbertsonl, Mark R. Miller1, Alex Olwal, Sean Follmer1 Stanford University 2Google Inc. *Grabity: A Wearable Haptic Interface for Simulating Weight and Grasping in Virtual Reality*
- [B7] *Augmented Reality* (PDF). Zums.ac.ir. Retrieved 19 April 2019
- [B8] *Vision based hand gesture recognition for human computer interactio*n: a survey SS Rautaray, A Agrawal - Artificial intelligence review, 2015 – Springer
- [B9] *Virtual Reality & Intelligent Hardware* Volume 1, Issue 2, April 2019
- [B10] M. B. Rhuby, R.A. Salguero and K. Holappa, *A Kalman Filtering tutorial for undergraduate Students*, International Journal of Computer Science and Engineering Survey, vol. 8 (1), 2017, pp. 1-18

Διαδικτυακές πηγές

- [Δ1] https://en.wikipedia.org/wiki/Haptic_technology
- [Δ2] https://www.robotics.org/blog-article.cfm/Robot-Hand-and-Haptic-Technology-Enables-a-New-Level-of-Human-Robot-Collaboration/216
- [Δ3] https://www.smithsonianmag.com/innovation/heres-what-future-haptic-technology-looks-or-rather-feels-180971097/
- [Δ4] https://en.wikipedia.org/wiki/Machine_vision
- [Δ5]https://www.electronicdesign.com/technologies/components/article/21802172/piezo-disks-deliver-vertical-haptic-feedback
- [Δ6] : http://eprints.utar.edu.my/3055/1/fyp_IA_2018_CYX_-_1403243.pdf
- [Δ7] https://www.roboticstomorrow.com/article/2020/03/cut-from-the-same-cloth-automatic-3d-recognition-and-marking-of-wooden-beams/15050/
- [Δ8] https://docs.opencv.org/
- [Δ9] https://www.kalmanfilter.net/kalman1d.html
- [Δ10] https://medium.com/analytics-vidhya/hand-detection-and-finger-counting-using-opencv-python-5b594704eb08