# RC CAR WITH COMPUTER VISION

Ανδρέας Βάλβης

# ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

# ΣΚΟΠΟΣ

Σκοπός της εργασίας είναι η έρευνα και η ανάπτυξη ενός ρομποτικού οχήματος που μπορεί να αλληλοεπιδράσει με ένα τελείως διαφορετικό τρόπο απ' ότι έχει συνηθίσει το κοινωνικό σύνολο. Η υπολογιστική όραση ήταν η μέθοδος που χρησιμοποιήθηκε δίνοντας στο ρομποτικό όχημα την ελευθερία της αυτονομίας από τα καλώδια και την ευκολία στο χειρισμό του. Ως αντικείμενο θα ενταχθεί στον τομέα της ψυχαγωγίας σαν ένα remote controlled car αλλά ως ιδέα έχει ευρύ φάσμα χρήσης σε εφαρμογές με κάποιες τροποποιήσεις στον κώδικα καθώς και με μία νέα διάταξη ρομποτικού οχήματος.

# ΘΕΩΡΗΤΙΚΟ ΜΕΡΟΣ

## Αυτόνομα κινούμενα ρομπότ - Τηλερομπότ

Η έννοια του ρομπότ είναι μια έννοια αμφιλεγόμενη κατά καιρούς , γενικότερα όμως σαν ορισμός ισχύει ότι πρέπει να είναι μια ηλεκτρονική-μηχανική διάταξη επαναπρογραμματιζόμενη, πολυλειτουργική και σχεδιασμένη για την διευκόλυνση των ανθρώπινων αναγκών. Υπάρχουν αρκετά κριτήρια διάκρισης και αντίστοιχες κατηγοριοποιήσεις για τα ρομπότ:
Ορισμένα κριτήρια είναι :

- Απευθείας έλεγχος του ρομπότ από έναν άνθρωπο που λέγεται ελεγκτής.
- Αυτόνομος έλεγχος υπό τον έλεγχο υλικολογισμικού.

Ορισμένες κατηγορίες είναι :

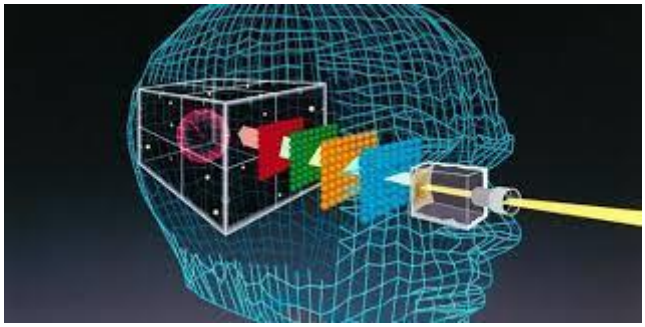
- Βιομηχανικά ρομπότ
- Κινητά ρομπότ
- Ιατρικά ρομπότ
- Τηλερομπότ

Το ρομποτικό όχημα που σχεδιάστηκε στο παρόν Project ανήκει στα Τηλερομπότ, τα οποία είναι ένας κινούμενος μηχανισμός , ελεγχόμενος ασύρματα από τον χειριστή μέσω Η/Υ είτε μέσω τηλεχειριστηρίου για την κίνηση του στον χώρο. Η σύνδεση του γίνεται με Wi-Fi είτε με Bluetooth και μπορεί να εκτελέσει τις κινήσεις δεξιά, αριστερά, πίσω και εμπρόσθια . Επίσης αν είναι εξοπλισμένο με αισθητήρες μπορεί να κάνει λήψη δεδομένων μέσω τηλεχειρισμού είτε λήψη εικόνας και να τα απεικονίζει σε Live μετάδοση στον χειριστή .

## Computer vision

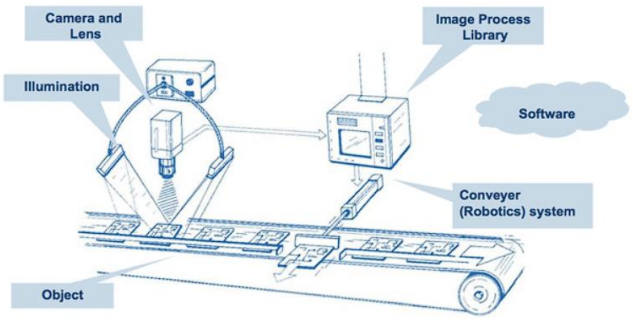
Computer vision είναι ο κλάδος της τεχνητής νοημοσύνης (AI) το οποίο επιτρέπει στο σύστημα/ρομπότ είτε υπολογιστή να χρησιμοποιεί την κάμερα του σαν μέσο χαρτογράφησης αντικειμένων, βίντεο είτε οποιοδήποτε άλλο εικονικό μέσο. Με το computer vision το σύστημα αποκτάει νοημοσύνη και σκέφτεται πώς να αντιδράσει με βάση το αντικείμενο που αναγνωρίζει.

Η υπολογιστική όραση είναι σαν την ανθρώπινη όραση, ο άνθρωπος έχει μάθει να ξεχωρίζει αντικείμενα και χειρονομίες από μικρή ηλικία έτσι ώστε με τον καιρό να έχει την εμπειρία να αναγνωρίζει τα αντικείμενα και τις εκάστοτε χειρονομίες. Στην υπολογιστική όραση Cv αντί για εγκέφαλο υπάρχουν οι νευρώνες που εκπαιδεύονται σαν ένα νευρωνικό δίκτυο ώστε να μαθαίνει νέα αντικείμενα και νέες χειρονομίες. Για την απλοποίηση του κειμένου θα ορίσουμε ως αντικείμενο τις χειρονομίες τις κινήσεις αλλά κ τα εικονικά ερεθίσματα που λαμβάνει ο άνθρωπος και η μηχανή.



Ένα προτέρημα του ανθρώπου είναι να αναγνωρίζει αντικείμενα από απόσταση και να καταλαβαίνει το βάθος οπότε να ξεχωρίζει αν το ίδιο αντικείμενο που βρίσκεται σε απόσταση 10 cm είναι το ίδιο σε απόσταση 300 cm, το οποίο οι κάμερες δεν είχαν καταφέρει για ένα μεγάλο διάστημα με αποτέλεσμα να δυσλειτουργούν. Πλέον η τεχνολογία έχει εξελιχθεί και έχουν μπει στο εμπόριο κάμερες με αισθητήρες βάθους, νυχτερινής όρασης κ.α. κάνοντας τα ρομπότ και τα μηχανήματα ακόμα πιο «έξυπνα».

Το Cv μαζί με τα νευρωνικά δίκτυα έχει εξελιχθεί, οπότε μαζί τους και η παραγωγή στις βιομηχανίες. Το Cv χρησιμοποιείται σε μια ποικιλία βιομηχανικών διεργασιών, όπως η επιθεώρηση υλικού, η αναγνώριση αντικειμένων, η αναγνώριση προτύπων, η ανάλυση ηλεκτρονικών εξαρτημάτων και οπτικών χαρακτήρων σε ταχύτητες milli second, πράγμα το οποίο καθιστά το cv ένα από τα top hot subjects για τις βιομηχανίες το 2022. Στην εικόνα 1 φαίνεται μία σειρά διεργασιών σε ένα σύγχρονο εργοστάσιο/βιομηχανία.




Σύστημα Machine Vision (Wikimedia)

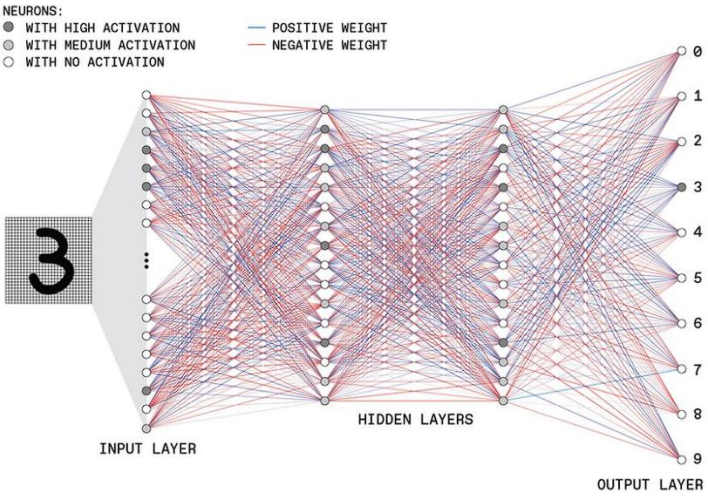
*Εικόνα 1 Σειρά διεργασιών στη βιομηχανία*

## Python

Η Python το 2022 είναι μία από τις πιο διαδεδομένες γλώσσες προγραμματισμού ευρύτερα για την τεχνητή νοημοσύνη και την μηχανική μάθηση. Χρησιμοποιεί interpreter και όχι compiler και ορίζεται σαν γλώσσα γενικού σκοπού. Η Python χρησιμοποιεί μια σημαντική εσοχή στο σχεδιασμό της, δίνοντας έμφαση στην αναγνωσιμότητα του κώδικα. Η γλώσσα εφαρμόζει μια αντικειμενοστραφή προσέγγιση για να επιτρέψει στους προγραμματιστές να γράφουν λογικό, καθαρό κώδικα τόσο για μικρά όσο και για μεγάλα projects.
Δύο βασικές τεχνολογίες που χρησιμοποιούνται για να επιτευχθεί το CV είναι:

- Ένας τύπος μηχανικής μάθησης που ονομάζεται βαθιά μάθηση
- Ένα συνελικτικό νευρωνικό δίκτυο (CNN).

Η βαθιά μάθηση (Deep Learning ,DL) είναι μία τεχνική που χρησιμοποιεί απλές μονάδες νευρώνων και τις συνθέτει μία μία για να δημιουργήσει ένα δίκτυο. Ο αριθμός των συνελίξεων μπορεί να είναι σημαντικά αυξημένος και τα χαρακτηριστικά διανύσματα που προκύπτουν να έχουν ελαττωμένες τις διαστάσεις σε μεγάλο ποσοστό, χωρίς όμως να χάνουν την ουσία της πληροφορίας που περιλαμβάνουν. Η είσοδος διέρχεται διαδοχικά σε μία προς μία στιβάδες νευρώνων και ο όρος «βαθύ» αναφέρεται στις κρυφές στοιβάδες των νευρώνων. Χρησιμοποιείται σε σύνθετα προβλήματα που υπάρχουν λίγα δεδομένα σαν είσοδοι. Ένα παράδειγμα DL είναι αυτό που φαίνεται στην επόμενη εικόνα (εικόνα 2) όπου αναλύεται ο αριθμός **3**. Από το input παίρνει είσοδο τα ορίσματα που έδωσε ο προγραμματιστής προκειμένου να διαχωρίσει το δίκτυο, διδάσκει τον εαυτό του σχετικά με το πλαίσιο των οπτικών δεδομένων, οπότε η πληροφορία πηγαίνει στα Hidden layers, όπου και αναλύεται με αλγόριθμους ώστε στην έξοδο να λαμβάνονται πληροφορίες. Έπειτα από αλγεβρική επεξεργασία προκύπτει αυτό που πρόβλεψε το δίκτυο.

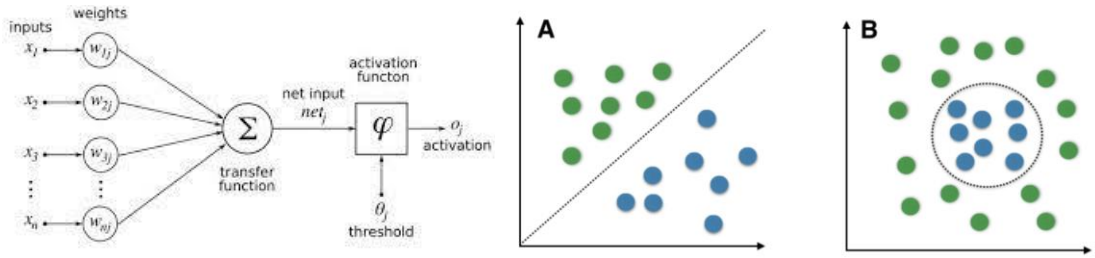



*Εικόνα 2 Νευρωνικό δίκτυο*

Σε αντίθετη πλευρά στο συνελικτικό νευρωνικό δίκτυο ή αλλιώς shallow χρησιμοποιούνται τα εξής βήματα :

1. Συνέλιξη της εισόδου με φίλτρα από την διαδικασία εκμάθησης με την μέθοδο back propagation
2. Εφαρμογή μη γραμμικότητας
3. Pooling
4. Κανονικοποίηση

Αυτό το νευρωνικό δίκτυο είναι πιο απλό σε σύγκριση με το προηγούμενο και αφορά γραμμικά προβλήματα διαχωρισμών κλάσεων , όπως διαχωρισμό των 2 δεδομένων.



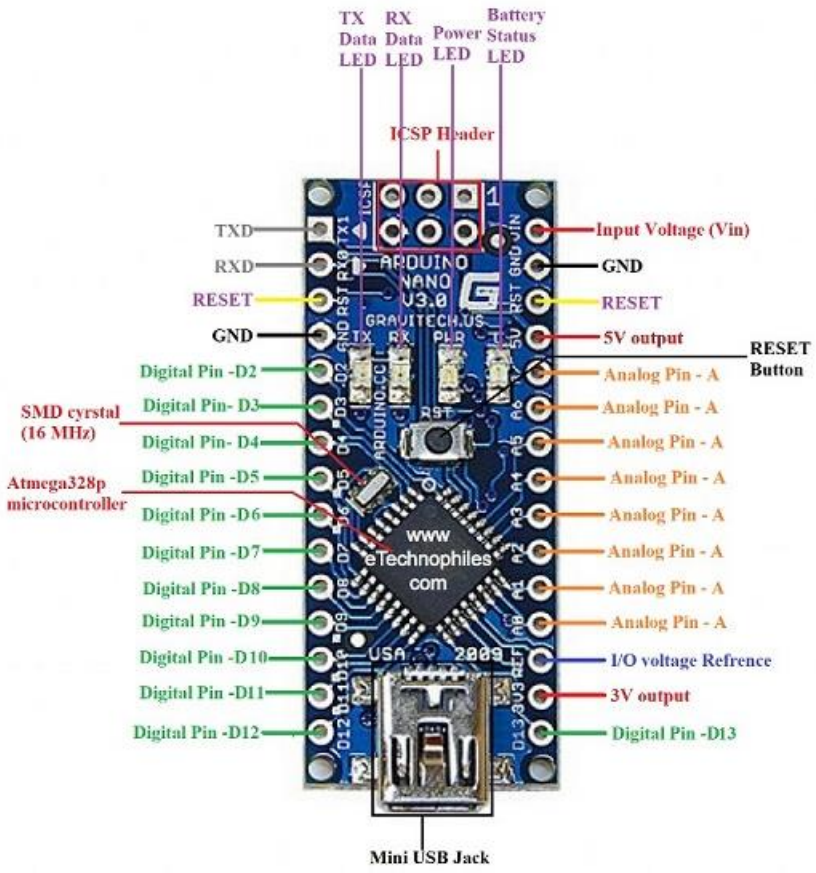

*Εικόνα 3 Αναγνώριση προτύπων*

Στο μέρος του κώδικα της Python που γράφτηκε, η mediapipe είναι η λύση για το νευρωνικό δίκτυο που χρειάζεται να δημιουργηθεί. Χρησιμοποιεί Machine learning αλγόριθμους βαθιάς μάθησης με ήδη εκπαιδευμένα μοντέλα, οπότε όταν ζητείται σαν solution τον mp hands γνωρίζει το ήδη εκπαιδευμένο δίκτυο να επιστρέψει την αναγνώριση του χεριού και την τοποθέτηση των Landmarks.

## Arduino

Η Arduino είναι μια πλατφόρμα ηλεκτρονικών ανοιχτού κώδικα, που περιλαμβάνει hardware και software εύκολα και φιλικά προς τον χρήστη. Τα αναπτυξιακά της arduino ποικίλουν, όμως όλα λειτουργούν με την ίδια λογική, όπου λαμβάνουν ερεθίσματα από το περιβάλλον μέσω αισθητήρων και έπειτα επιδρούν σε αυτό, ελέγχοντας διάφορες εξόδους, όπως leds, κινητήρες, ή άλλους ενεργοποιητές. Η Arduino προσφέρει την δικιά της πλατφόρμα προγραμματισμού στην οποία συντάσσεται ο κώδικας για των προγραμματισμό των αναπτυξιακών της και έπειτα φορτώνεται σε αυτά μέσω θύρας USB.

Το αναπτυξιακό που επιλέχθηκε για την υλοποίηση του project είναι το **Arduino Nano**. Επιλέχθηκε κυρίως λόγω του μεγέθους του, καθώς είναι μικρό και εύκολο να τοποθετηθεί μέσα στην κατασκευή. Να σημειωθεί ότι αποτελεί την μικρογραφία του Arduino Uno, αφού δεν υστερεί σε ποσότητα διαθέσιμων pins εισόδων – εξόδων. Ο μικροελεγκτής του είναι ο ATmega328, ένας επεξεργαστής 8-bit που μπορεί να επιτύχει έως 16 MIPS (Million Instructions Per Second) για 16 MHz clock frequency. Έχει διαθέσιμα 32 KB, εκ των οποίων τα 2 KB χρησιμοποιούνται για τον bootloader, 2 KB εσωτερική SRAM και 1 KB EEPROM. Το Arduino Nano προσφέρει 22 digital pins, 8 analog pins, 6 PWM outputs, όπως επίσης και 1 θύρα USB Mini-B.




*Εικόνα 4 Arduino Nano Pinout*

Η γλώσσα προγραμματισμού της Arduino είναι βασισμένη στην C/C++. Ένα πρόγραμμα γραμμένο στο περιβάλλον της Arduino ονομάζεται sketch και έχει κατάληξη αρχείου **.ino**. Κάθε πρόγραμμα αποτελείται από τουλάχιστον 2 βασικές συναρτήσεις: την **setup()**, η οποία τρέχει μία μόνο φορά στην αρχή του προγράμματος και την **loop()**, η οποία τρέχει κατ' επανάληψη όσο τρέχει το πρόγραμμα. Βασικό πλεονέκτημα της γλώσσας είναι η πληθώρα ενσωματωμένων βιβλιοθηκών που παρέχονται, οι οποίες επιτρέπουν την εύκολη πρόσβαση στις λειτουργίες που προσφέρουν τα αναπτυξιακά της Arduino.

# ΠΡΑΚΤΙΚΟ ΜΕΡΟΣ

## Υλικά

Για την υλοποίηση του rc car χρειάστηκαν:

### Ηλεκτρονικά

- 1 Arduino Nano → Μικροεπεξεργαστής
- 1 L293D → Motor driver
- 1 HC -05 → Bluetooth module
- 2 DC gear motors → Κίνηση τροχών
- 4 μπαταρίες AAA → Τροφοδοσία Arduino Nano
- 1 μπαταρία 9V → Τροφοδοσία Μοτέρ

- Καλώδια → Συνδέσεις
- Καλώδιο USB → Ανέβασμα προγράμματος στο Arduino Nano
- Mini breadboard → Συνδεσμολογία
- Κουτάκι για τις μπαταρίες

### Κατασκευαστικά

- Κομμάτια 3D printed
- Βίδες
- Ταινία διπλής όψεως
- Τροχοί
- Φύλλα ξύλου μπάλσα
- Μονωτική ταινία
- Μεταλλικός άξονας
- Κόλλα δύο συστατικών

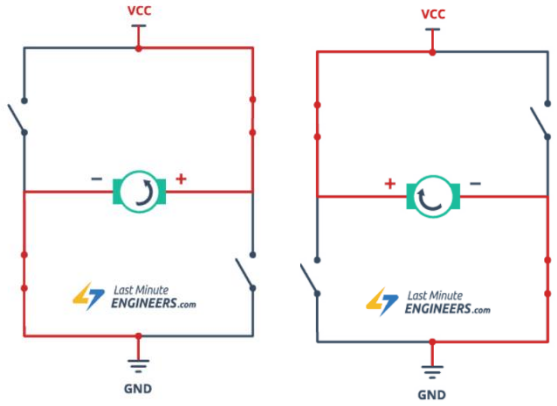
## Ανάλυση

### *L293D*

Το ολοκληρωμένο L293D είναι ένας motor driver που μπορεί να ελέγξει την ταχύτητα και την κατεύθυνση περιστροφής των κινητήρων. Προκειμένου να το επιτύχει αυτό, χρησιμοποιεί δύο τεχνικές :

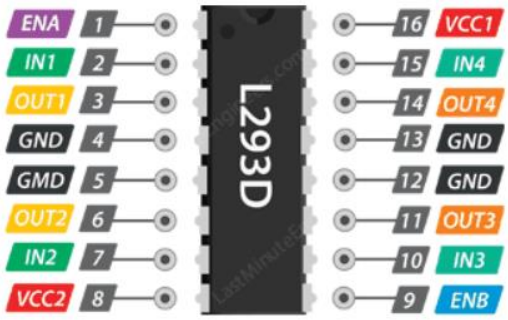
1. PWM (παλμό) για τον έλεγχο της ταχύτητας (pins ENA, ENB)
   Όσο πιο μεγάλο είναι το duty cycle, τόσο μεγαλύτερη είναι η μέση τάση που εφαρμόζεται στα άκρα του κινητήρα – επομένως και η ταχύτητα περιστροφής είναι μεγαλύτερη – και αντίστοιχα όσο πιο μικρό είναι το duty cycle, τόσο μικρότερη είναι η μέση τάση που εφαρμόζεται στα άκρα του κινητήρα – επομένως και η ταχύτητα περιστροφής είναι μικρότερη.

2. H – bridge για τον έλεγχο της κατεύθυνσης περιστροφής (pins IN1, IN2, IN3, IN4)




Το κύκλωμα της γέφυρας αποτελείται από 4 διακόπτες και τον κινητήρα να βρίσκεται στη μέση όπως φαίνεται στη διπλανή εικόνα (εικόνα 5). Όταν οι δύο από τους τέσσερις διακόπτες κλείνουν με συγκεκριμένο τρόπο ταυτόχρονα, τότε αντιστρέφεται η πολικότητα της τάσης που εφαρμόζεται στα άκρα του κινητήρα και αλλάζει η κατεύθυνση περιστροφής του.

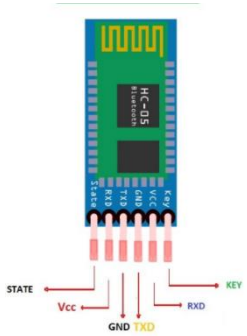
*Εικόνα 5 H - bridge*




*Εικόνα 6 L293D pinout*

## *HC – 05*

Το HC - 05 είναι ένα Bluetooth SPP (Serial Port Protocol) module V2.0+EDR (Βελτιωμένος ρυθμός δεδομένων) 3Mbps. Έχει -80dBm ευαισθησία και +4dBm RF δυνατότητα εκπομπής και το default baud rate είναι τα 38400. Η σύνδεσή του γίνεται με τον υπολογιστή στον οποίο τρέχει το πρόγραμμα της python για το computer vision. Έχει 6 pins εκ των οποίων τα χρησιμοποιούμενα είναι τα 4:

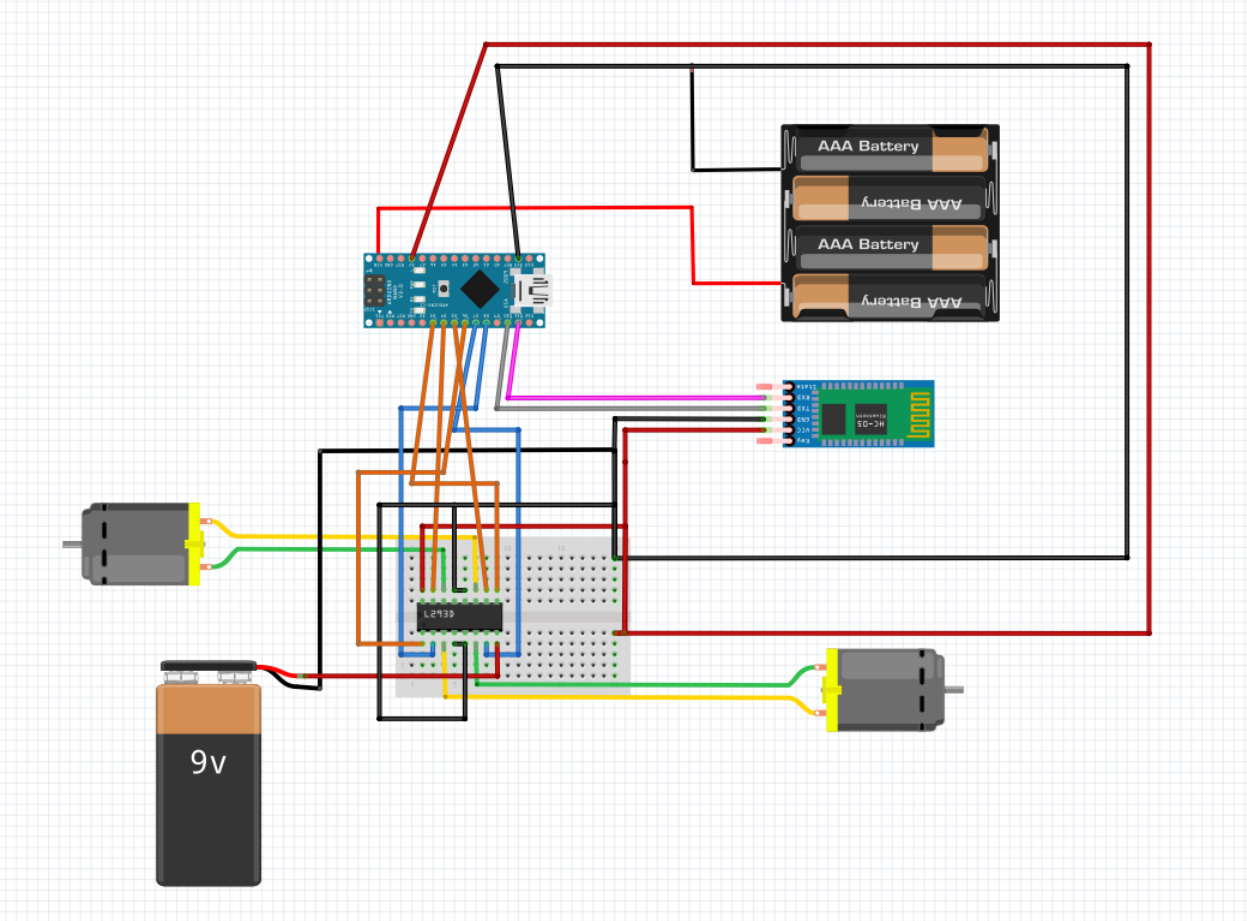



- Vcc: τροφοδοσία
- Gnd: γείωση
- Rx: μετάδοση δεδομένων
- Tx: μετάδοση δεδομένων

*Εικόνα 7 HC-05 pinout*

## Συνδεσμολογία

Η συνδεσμολογία του ηλεκτρονικού κυκλώματος είναι η παρακάτω:




*Εικόνα 8 Συνδεσμολογία κυκλώματος*

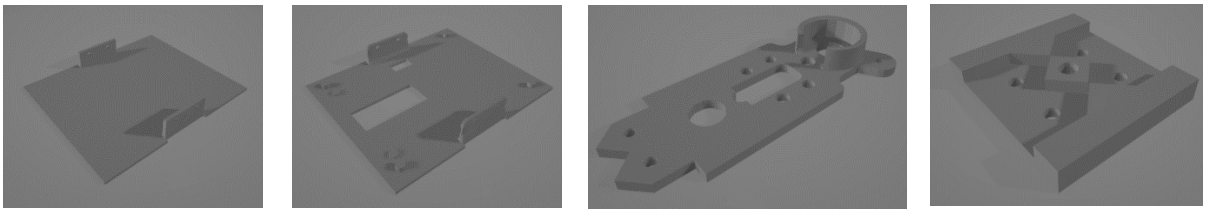
| L293D | ARDUINO NANO | L293D | ARDUINO NANO |
|---|---|---|---|
| 1 | D6 | 9 | D3 |
| 2 | D8 | 10 | D5 |
| 3 | MOTOR A | 11 | MOTOR B |
| 4 | GND | 12 | GND |
| 5 | GND | 13 | GND |
| 6 | MOTOR A | 14 | MOTOR B |
| 7 | D7 | 15 | D4 |
| 8 | 9V | 16 | 5V |

| HC – 05 | ARDUINO NANO |
|---|---|
| VCC | 5V |
| GND | GND |
| RX | D11 |
| TX | D10 |

Να σημειωθεί ότι το όχημα έχει κίνηση μόνο στου πίσω τροχούς, διότι τα μοτεράκια είναι δύο και όχι τέσσερα.

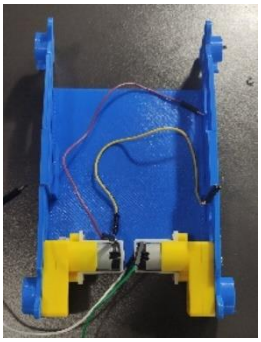
## 3D print – συναρμολόγηση

Αρχικά, για τον σκελετό της κατασκευής χρησιμοποιήθηκαν εκτυπωμένα κομμάτια από 3D εκτυπωτή. Το ολοκληρωμένο σχέδιο αποτελείται από 6 κομμάτια εκ των οποίων χρησιμοποιήθηκαν τα 4 από αυτά. Το υλικό εκτύπωσης είναι το PLA.



*Εικόνα 9 3D εκτυπωμένα κομμάτια*

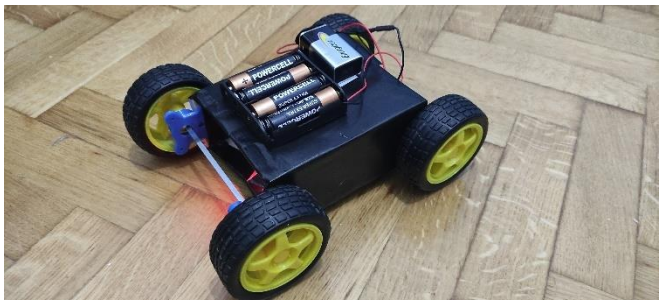
Τα κομμάτια συνδέθηκαν μεταξύ τους με βίδες, όπως επίσης και τα μοτεράκια για την κίνηση στους πίσω τροχούς. Οι μπροστινοί τροχοί συνδέθηκαν μεταξύ τους με μεταλλικό άξονα ο οποίος στερεώθηκε στις εσοχές των τροχών με κόλλα 2 συστατικών, ενώ για τους πίσω τροχούς δεν ήταν απαραίτητο κάτι τέτοιο, αφού τα μοτεράκια είναι πλήρως συμβατά με τους συγκεκριμένους τροχούς. Το arduino nano και το HC -05 τοποθετήθηκαν στις ειδικές εσοχές που έχει το 3d σχέδιο, ενώ το breadboard και οι μπαταρίες κολλήθηκαν με ταινία διπλής όψεως πάνω στην εκτυπωμένη βάση και στην οροφή του οχήματος αντίστοιχα.



*Εικόνα 11 Στάδιο συναρμολόγησης*

*Εικόνα 10 Τελικό στάδιο*

Εξ’ αιτίας ενός προβλήματος, που θα αναφερθεί στην αντίστοιχη ενότητα, χρειάστηκε να ξαναδημιουργηθεί ο σκελετός του οχήματος. Αυτή τη φορά επιλέχθηκε ως υλικό φύλλα ξύλου μπάλσας πάχους 1 cm. Οι διαστάσεις παρέμειναν ίδιες, παίρνοντας ως πρότυπο τα προηγούμενα σχέδια για τον σκελετό και η συναρμολόγηση των κομματιών έγινε με μαύρη μονωτική ταινία. Τα ηλεκτρονικά εξαρτήματα, όπως και προηγουμένως στερεώθηκαν στο όχημα με ταινία διπλής όψεως.



*Εικόνα 12 Τελικό στάδιο - Νέος σκελετός*

## Κώδικες

### Κώδικας python

Για το κομμάτι της python χρησιμοποιήθηκε το πρόγραμμα PyCharm. Ο κώδικας αποτελείται από δύο script:

*Script 'Main.py'*

```
import cv2
import mediapipe as mp
import time
import controller as cnt
```

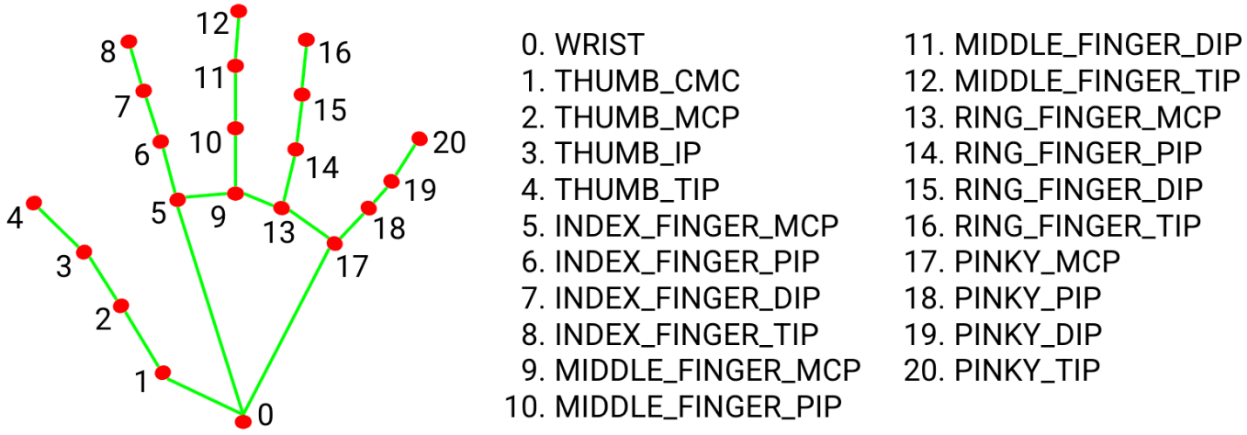
Δηλώνονται οι βιβλιοθήκες για το computer vision όπου θα αντιμετωπιστεί η επίλυση προβλημάτων υπολογιστικής όρασης και επεξεργασίας εικόνας. Το mediapipe είναι ο αλγόριθμος του νευρωνικού δικτύου από που πηγάζει το Hand Landmark model άρα και οι συναρτήσεις που σχεδιάζουν το μοντέλο του χεριού .Η time χρησιμοποιείται για να δώσει delay ώστε να σταματήσει την λειτουργία του προγράμματος για καθορισμένο χρονο και το controller στο script της Python μπορεί να επηρεάσει και άλλα scripts μέσα στον ίδιο φάκελο από το οποίο θα πηγάζει ,αρκεί στο αρχικό sketch να δηλωθεί ο controller και οι υπορουτίνες που θα εκτελεί.

```
time.sleep(2.0)

mp_draw = mp.solutions.drawing_utils
mp_hand = mp.solutions.hands
```

Το time δηλώθηκε 2 sec γιατί χρειαζόταν να τρέξουν πιο γρήγορα τα υπόλοιπα προγράμματα που βρίσκονταν στον ίδιο φάκελο διότι διαπιστώθηκε πρόβλημα με την επικοινωνία του Bluetooth και την επεξεργασία των δεδομένων. Στην συνέχεια, δημιουργούνται δύο συναρτήσεις που βοηθούν στην σχεδίαση διαφορετικών χειρονομιών: η πρώτη ανιχνεύει το χέρι και η δεύτερη γνωρίζει να το σχεδιάζει όπως φαίνεται στην εικόνα 10 παρακάτω.

```
tipIds = [4, 8, 12, 16, 20]
```

Εδώ δημιουργείται μια λίστα μόνο με τα άκρα των δακτύλων, γιατί με βάση αυτά θα αναγνωρίζεται εάν ένα δάκτυλο είναι προς τα πάνω ή προς τα κάτω.




*Εικόνα 13 Σχεδίαση παλάμης*

```
video = cv2.VideoCapture(0)
```

Ανοίγει η default webcam του υπολογιστή.

```
with mp_hand.Hands(min_detection_confidence=0.5,
                   min_tracking_confidence=0.5) as hands:
```

Το ποσοστό αναγνώρισης αλλά και εντοπισμού παλάμης ορίζεται στο 0,5, διότι είναι η καλύτερη τιμή, εφόσον όσο αυξάνεται θα υπάρξουν καλύτερα αποτελέσματα εντοπισμού και αναγνώρισης αλλά και σημαντική χρονική καθυστέρηση στην μετάδοση των αποτελεσμάτων από το image frame στις συναρτήσεις του processing.

```
 while True:
ret, image = video.read()
```

Στο σημείο αυτό ξεκινάει ένας βρόχος επανάληψης που θα σταματήσει μόνο όταν έρθει μια εντολή *brake*. Μέχρι τότε δημιουργείται το *ret* που είναι μία Boolean μεταβλητή και επιστρέφει άσσο (1) όταν το frame της κάμερας είναι ανοικτό και το image είναι ένα διάνυσμα πίνακα εικόνων που λαμβάνεται με βάση τα προεπιλεγμένα καρέ ανά δευτερόλεπτο.

```
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

Γίνεται αλλαγή διάταξης χρωμάτων εικόνας από BLUE GREEN LED σε RED GREEN BLUE ως ένα βήμα για την αντιμετώπιση προβλήματος επεξεργασίας εικόνας, αφότου το OpenCv έχει by default άλλη διάταξη χρωμάτων όταν πρόκειται να επεξεργαστεί μία εικόνα.

```
image.flags.writeable = False
```

Αποκλεισμός της καταγραφής περιεχομένου για την καλύτερη απόδοση του προγράμματος αλλά και σε περίπτωση που θελήσει κάποιο άλλο μέσο να χρησιμοποιήσει την κάμερα , δεν θα του δίνεται η πρόσβαση.Η κατάσταση των δεδομένων της εικόνας είναι μόνο readable.

```
results = hands.process(image)
```

Σαν πρώτο βήμα, χρησιμοποιούμε τη συνάρτηση process από τη βιβλιοθήκη Mediapipe για να αποθηκευτούν τα αποτελέσματα ανίχνευσης «κουκίδων» του χεριού στη μεταβλητή results. Σε αυτό το σημείο γίνεται η επεξεργασία των readable δεδομένων.

```
image.flags.writeable = True
```

Όταν τελειώσει η επεξεργασία τα δεδομένα επανέρχονται σε δυναμικά και αλλάζουν ανάλογα το καρέ/sec.

```
image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
```

Αφού τελειώσει η επεξεργασία γίνεται επιστροφή στα χρώματα που βλέπουν οι άνθρωποι στην καθημερινότητα.

```
lmList = []
```

Δημιουργείται μία λίστα που θα γεμίζει ανάλογα με το ID του δακτύλου.

```
# check wether any landmark was detected
if results.multi_hand_landmarks:
 for hand_landmark in results.multi_hand_landmarks:
 #Which hand are we talking about
        myHands = results.multi_hand_landmarks[0]
```

Όταν εντοπιστούν κουκίδες στο χέρι από το *Hand process*, δημιουργείται μία επανάληψη, για να συμπληρωθούν κάποιοι πίνακες που θα αναφερθούν στην συνέχεια, έτσι ώστε τα αποτελέσματα της επεξεργασίας σε συνδυασμό με το πλήθος των δακτύλων (συνάρτηση multi) να αποθηκεύονται σε ένα πίνακα που θα έχει μέσα του πίνακες. Δηλαδή, ένας πίνακας με τα δάκτυλα που θα αναγνωρίζει και κάθε δάκτυλο θα έχει δικό του πίνακα με τα σημεία του στο χώρο, του οποίου η υλοποίηση φαίνεται παρακάτω.

```
# vres to id tou daktulou kai landmark info
for id, lm in enumerate(myHands.landmark):
# height width kai channel
                    h, w, c = image.shape
#vres thn 8esh tou daktulou
                    cx, cy = int(lm.x * w), int(lm.y * h)
# print(id,cx,cy)
                    lmList.append([id, cx, cy])
#sxediase kyklakia gia ta points pou vrhkes
                                    mp_draw.draw_landmarks(image,   hand_landmark,
mp_hand.HAND_CONNECTIONS)
```

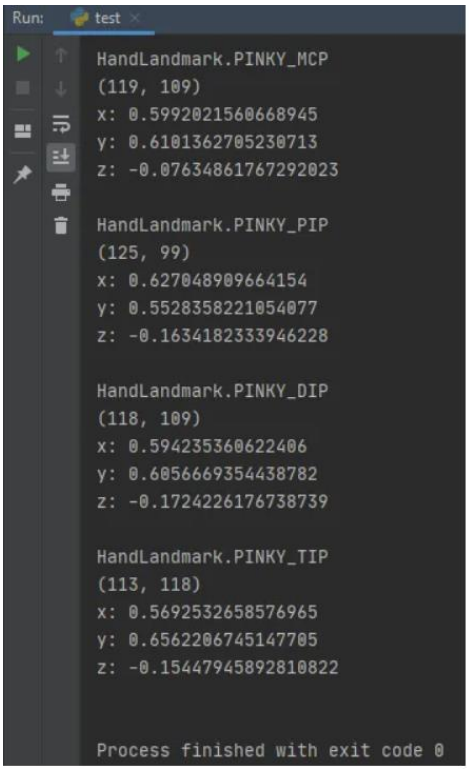
Υπολογίζεται το Coordinates X, Coordinates Y (cx , cy) κάθε δακτύλου με τον εξής τρόπο:, ορίζονται τα X, Y που επιστρέφει η συνάρτηση *shape* πολλαπλασιασμένο με το *width* και *height* αντίστοιχα, ώστε να επιστραφούν οι θέσεις του δακτύλου στον χώρο .Ύστερα επιλέγονται οι τιμές με το *lm.list.append([])* και σχεδιάζονται τα κυκλάκια πάνω στις τιμές.

Ένα κομμάτι κώδικα για debugging ήταν το παρακάτω , το οποίο κάθε φορά έδειχνε ως αποτέλεσμα τον πίνακα με τους εσωτερικούς πίνακες. Με παρόμοια λογική υπολόγιζε τις κανονικοποιημένες συντεταγμένες X, Y . Με την κανονικοποίηση των μεταβλητών, είναι εφικτό να παρατηρηθεί εάν ένα σύνολο μετρούμενων μεταβλητών μετράει πραγματικά το ίδιο πράγμα και στην πραγματικότητα δεν υπάρχουν λανθασμένα μεγέθη λόγω θορύβου, όπως και δημιουργείται αν το χέρι κινείται πολύ γρήγορα στην κάμερα κατά την λήψη δεδομένων.

```
with handsModule.Hands(static_image_mode=True) as hands:

    ret, image = video.read()
    results = hands.process(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))

    imageHeight, imageWidth, _ = image.shape

    if results.multi_hand_landmarks != None:
        for handLandmarks in results.multi_hand_landmarks:
          for point in handsModule.HandLandmark:

              normalizedLandmark = handLandmarks.landmark[point]
```

```
                                                    pixelCoordinatesLandmark      =
drawingModule._normalized_to_pixel_coordinates(normalizedLandmark.x,
normalizedLandmark.y, imageWidth, imageHeight)

            print(point)
            print(pixelCoordinatesLandmark)
            print(normalizedLandmark)
```

Αυτό το κομμάτι επέστρεφε τις πληροφορίες από τα Landmarks που υπολόγιζαν οι συναρτήσεις. Στην αρχή υπήρχε αυτός ο κώδικας, αλλά μπήκε ο προηγούμενος κώδικας γιατί εμπειρικά έδειξε ότι ήταν πιο γρήγορος, στον υπολογισμό των πληροφοριών των θέσεων των δακτύλων αργούσε αρκετά η διαδικασία. Σαν debugging κώδικας ήταν αρκετό.




Εδώ είναι οι πληροφορίες που επέστρεφε η
Print(point)
Print(PixelCoordinatesLandmark)
Print(normalizedLandmark)
Το παρόν όπως και το προηγούμενο κομμάτι δεν ανήκουν πλέον στον τωρινό κώδικα (για αυτό και δεν θα αναλυθούν). Αφαιρέθηκαν διότι αρχικά ήταν αργά υπολογιστικά και επειδή δεν ήταν δυνατό να βγει μία λίστα ή ένας πίνακας με τα συνολικά δεδομένα, ώστε να αποσταλθεί στο Arduino για την λήψη αποφάσεων.

Συνεχίζοντας τον κώδικα του project :

```
if len(lmList) != 0:
            if lmList[tipIds[0]][1] > lmList[tipIds[0] - 1][1]:
                fingers.append(1)
            else:
                fingers.append(0)
```

Μία έτοιμη λύση από την ComputerVisionZone είναι να τοποθετήσει κάποια σημεία στο δάκτυλο 1 (αντίχειρας). Εάν το σημείο του άκρου έχει τιμή μικρότερη από το ακριβώς από κάτω σημείο (-1) τότε append(1) δηλαδή αντίχειρας κατεβασμένος.

```
for id in range(1, 5):
                if lmList[tipIds[id]][2] < lmList[tipIds[id] - 2][2]:
                    fingers.append(1)
                else:
                    fingers.append(0)
            total = fingers.count(1)
```

Για τα υπόλοιπα δάκτυλα είναι ακριβώς η ίδια λογική αλλάζει όμως η σύγκριση , γίνεται με δύο σημεία πιο κάτω , όχι ένα.

```
cnt.motor(total)
```

Η συνάρτηση του control η οποία μοιράζει στα υπόλοιπα προγράμματα στον ίδιο φάκελο τις πληροφορίες του τωρινού script.

```
if total == 0:
              cv2.rectangle(image, (20, 300), (270, 425), (0, 0, 255), cv2.FILLED)
               cv2.putText(image, "stop", (45, 375), cv2.FONT_HERSHEY_SIMPLEX,
                           2, (0, 0, 0), 5)
```

Εάν *δάκτυλα =0,* δηλαδή όλα κατεβασμένα εμφανίζεται ένα image παραλληλογράμμου με διαστάσεις (270,425) και με την τοποθέτηση των γραμμάτων να γίνεται σε διάσταση (20,300) και το χρώμα να είναι Κοκόκκινο (BGR). Μετά εμφανίζεται ένα κείμενο *text = stop* με διαστάσεις (45,375) font 2 σαν γραμματοσειρά, χρώμα (*0,0,0 = black*) και 5 για το Bold.

```
elif total == 1:
              cv2.rectangle(image, (20, 300), (270, 425), (0, 255, 0), cv2.FILLED)
               cv2.putText(image, "right", (45, 375), cv2.FONT_HERSHEY_SIMPLEX,
                           2, (0, 0, 0), 5)

           elif total == 2:
              cv2.rectangle(image, (20, 300), (270, 425), (0, 255, 0), cv2.FILLED)
               cv2.putText(image, "left", (45, 375), cv2.FONT_HERSHEY_SIMPLEX,
                           2, (0, 0, 0), 5)

           elif total == 3:
              cv2.rectangle(image, (20, 300), (270, 425), (0, 0, 255), cv2.FILLED)
               cv2.putText(image, "stop", (45, 375), cv2.FONT_HERSHEY_SIMPLEX,
                           2, (0, 0, 0), 5)

           elif total == 4:
              cv2.rectangle(image, (20, 300), (270, 425), (0, 0, 255), cv2.FILLED)
               cv2.putText(image, "stop", (45, 375), cv2.FONT_HERSHEY_SIMPLEX,
                           2, (0, 0, 0), 5)

           elif total == 5:
              cv2.rectangle(image, (20, 300), (270, 425), (0, 255, 0), cv2.FILLED)
```

```
                cv2.putText(image, "go", (45, 375), cv2.FONT_HERSHEY_SIMPLEX,
                            2, (0, 0, 0), 5)
```

Ακολούθησε ακριβώς η ίδια λογική και να σημειωθεί ότι πειραματικά μετρήθηκαν όλα τα images & texts

Στο επόμενο κομμάτι κώδικα δημιουργείται η εντολή συνθήκης τερματισμού του προγράμματος. Σημειώνεται ότι υπήρξε πρόβλημα κατά το κλείσιμο της εφαρμογής και τα frames του image show δεν έκλειναν, οπότε μετά από έρευνα στο διαδίκτυο ήρθε αυτή η λύση.

```
Cv2.imshow(“Frame”, image
X = cv2.waitKey(1)        # X for exit
if X == ord(‘x’) :
     Break
Video.release()
Cv2.destroyAllWindows()
```

Η waitkey παίρνει σαν όρισμα το 1 ώστε να ενεργοποιηθεί στο αμέσως επόμενο πάτημα πλήκτρου και από κάτω ορίζεται ποιο πλήκτρο εξυπηρετεί. Η Video release σταματάει να κάνει capture live μετάδοση εικόνας αφήνοντας τους πόρους της κάμερας ελευθέρους και η destroy τερματίζει ολοκληρωτικά κάθε παράθυρο που είναι ανοικτό στο πρόγραμμα.

*Script ‘Controller.py’*

```
import serial
```

Δηλώνεται η βιβλιοθήκη serial που επιτρέπει την πρόσβαση της σειριακής θύρας του υπολογιστή στο πρόγραμμα της Python για την εξαγωγή και εισαγωγή δεδομένων. Το Bluetooth ανήκει σε μία Backend πόρτα η οποία ήταν η COM7 στον υπολογιστή ελέγχου του παρόν Script.

```
ser = serial.Serial('COM7', 9600, timeout = None)
```

Δηλώνεται ποιά συριακή πόρτα θα χρησιμοποιηθεί, το baud rate και το timeout. Θα μεταφέρονται 9600 bits το δευτερόλεπτο, δεν χρησιμοποιήθηκε μεγαλύτερο baud rate γιατί το arduino μπορεί να διαβάσει έως ένα όριο δεδομένων, ύστερα μπλοκάρει την πόρτα για να μην γίνει υπερχείλιση δεδομένων. Το time out είναι για να προστατευτεί η υπερχείλιση δεδομένων αλλά και σε περίπτωση που δεν χρειάζεται να χρησιμοποιηθεί η πόρτα για περισσότερο από ένα διάστημα, στην περίπτωση του project δεν πρέπει να κλείσει γιατί γίνεται συνεχής λήψη δεδομένων.

```
def motor(total):
```

Σε αυτό το σημείο αναφέρεται η συνάρτηση του προηγούμενου script και ο υπόλοιπος κώδικας συντρέχει με το προηγούμενο script της Python

```
if total==0:    #stop moving
        ser.write(b'0')
```

Εάν η μεταβλητή total είναι 0 τότε στέλνει από το HC05 στο arduino byte 0, που ορίζεται σαν stop moving action. Ομοίως για τα υπόλοιπα :

```
    elif total==1:  #go right
        ser.write(b'1')

    elif total==2:  #go left
        ser.write(b'2')

    elif total==3:  #stop moving
        ser.write(b'0')

    elif total==4:  #stop moving
        ser.write(b'0')

    elif total==5:  #go forward
        ser.write(b'5')
```

## Κώδικας arduino

Ο προγραμματισμός του arduino nano έγινε στο προγραμματιστικό περιβάλλον της arduino. Ακολουθεί ανάλυση του κώδικα.

Αρχικά, γίνεται αρχικοποίηση βιβλιοθηκών. Η βιβλιοθήκη που χρησιμοποιήθηκε είναι η **SoftwareSerial** η οποία κάνει δυνατή την σειριακή επικοινωνία στα digital pins του arduino χρησιμοποιώντας λογισμικό. Είναι εφικτό να υπάρχουν πολλαπλές σειριακές πύλες με ταχύτητα έως και 11520 bps.

Έπειτα ορίζεται ότι το pin TX του HC05 αντιστοιχεί στο pin D10 του arduino nano και αντίστοιχα ότι το pin RX του HC05 αντιστοιχεί στο pin D11 του arduino nano, μέσω της εντολής **SoftwareSerial HC05**, η οποία παρέχεται από την βιβλιοθήκη.

```
1 #include <SoftwareSerial.h>
2 SoftwareSerial HC05(10, 11); //HC05-TX Pin 10, HC05-RX to Arduino Pin 11
```

*Εικόνα 14 Αρχικοποίηση βιβλιοθηκών*

Συνεχίζοντας τις αρχικοποιήσεις, ορίζονται οι μεταβλητές που θα χρησιμοποιηθούν για την σύνδεση του arduino nano στο ολοκληρωμένο L293D. Τα ονόματα των μεταβλητών επιλέχθηκαν να αντιστοιχούν στα ονόματα των pins του L293D (εικόνα 3) για πρακτικούς λόγους και οι μεταβλητές είναι τύπου int, διότι παίρνουν ακεραίους ως τιμές, οι οποίοι αντιπροσωπεύουν τα αντίστοιχα digital pins του arduino nano. Έτσι αναπαρίσταται η συνδεσμολογία και σε επίπεδο κώδικα.

```
// Motor A connections
int enA = 6;
int in1 = 8;
int in2 = 7;
// Motor B connections
int enB = 3;
int in3 = 5;
int in4 = 4;
```

*Εικόνα 15 Αρχικοποίηση μεταβλητών*

Ακολουθεί η πρώτη βασική συνάρτηση του προγράμματος, η **setup()**. Τρέχει μόνο μια φορά σε αυτό το σημείο του κώδικα και ορίζει την αρχική κατάσταση του οχήματος καθώς και βασικές παραμέτρους για τη λειτουργία του.

Ορίζεται το baud rate στα 9600 για το Bluetooth module HC05, ίδιο με το baudrate που έχει ορισθεί και στον κώδικα της python. Έτσι καθίσταται εφικτός ο συγχρονισμός και επομένως η μετάδοση δεδομένων. Επιλέχθηκε το 9600 διότι είναι το μικρότερο δυνατό που μπορεί να εξυπηρετήσει τις ανάγκες του προγράμματος.

Τα pins που ελέγχουν τα μοτέρ, δηλαδή οι μεταβλητές που αρχικοποιήθηκαν προηγουμένως, θέτονται σαν έξοδοι με την εντολή **pinMode**. Επίσης τα pins που ελέγχουν την κατεύθυνση περιστροφής θέτονται σε κατάσταση LOW, δηλαδή δεν θα περιστρέφονται οι τροχοί, ώστε όταν το όχημα βρίσκεται στην αρχική του κατάσταση να παραμένει ακίνητο.

```
void setup() {
  HC05.begin(9600); //Baudrate 9600

  // Set all the motor control pins to outputs
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);

  // Turn off motors - Initial state
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
}
```

*Εικόνα 16 Συνάρτηση setup()*

Το πρόγραμμα τελειώνει με τη δεύτερη βασική συνάρτηση, την **loop()**, η οποία τρέχει καθ' όλη τη διάρκεια που το όχημα τροφοδοτείται.

Στις πρώτες δύο σειρές λαμβάνονται δεδομένα από το HC05 με την εντολή **HC05.read** και αποθηκεύονται στην μεταβλητή τύπου character '*receive*'. Η *receive* είναι πιθανό να πάρει τις τιμές 0, 1, 2 και 5 καθώς αυτές είναι οι τιμές οι οποίες έχουν οριστεί το script *controller* της python

που θα στέλνει το bluetooth module HC05. Η εντολή *uint8_t* είναι το ίδιο με ένα byte, μια συντόμευση για ένα μη προσημασμένο ακέραιο με μήκος 8 bits.

Με την **analogWrite** τα pins που ελέγχουν την ταχύτητα των κινητήρων θέτονται σε 255, η οποία είναι η μέγιστη τιμή που μπορούν να πάρουν. Τα DC gear motors που χρησιμοποιούνται στην κατασκευή δίνουν την δυνατότητα ελέγχου της ταχύτητας περιστροφής τους. Η λειτουργία αυτή δεν ήταν απαραίτητη στην εφαρμογή, οπότε οι κινητήρες τέθηκαν σε πλήρη ταχύτητα καθ΄ όλη τη διάρκεια του προγράμματος.

Τέλος, υπάρχει μια δομή ελέγχου για την τιμή της μεταβλητής *receive* αποτελούμενη από αλλεπάλληλες εντολές **if**. Ανάλογα με την τιμή της μεταβλητής *receive* θέτονται οι ανάλογοι τροχοί σε κίνηση και με την ανάλογη φορά περιστροφής. Τα pins που επηρεάζουν την περιστροφή των κινητήρων έρχονται σε κατάσταση HIGH και LOW αναλόγως:

- *Receive =0* → το όχημα δεν κινείται
- *Receive =1* → το όχημα στρίβει δεξιά
- *Receive =2* → το όχημα στρίβει αριστερά
- *Receive =5* → το όχημα κινείται μπροστά

Να σημειωθεί ότι κατά το στρίψιμο οι κινητήρες περιστρέφονται με αντίθετες φορές με αποτέλεσμα το όχημα να τρίβει επί τόπου.

```
void loop() {
    uint8_t i;
    char receive = HC05.read(); //Read from Serial Communication

    analogWrite(enA, 255);
    analogWrite(enB, 255);

    if(receive == '5'){          //FORWARD ----
        digitalWrite(in1, HIGH);
        digitalWrite(in2, LOW);
        digitalWrite(in3, HIGH);
        digitalWrite(in4, LOW);
    }

    if(receive == '0'){          //STOP ---
        digitalWrite(in1, LOW);
        digitalWrite(in2, LOW);
        digitalWrite(in3, LOW);
        digitalWrite(in4, LOW);

    }

    if(receive == '2'){          //LEFT ---
        digitalWrite(in1, LOW);
        digitalWrite(in2, HIGH);
        digitalWrite(in3, HIGH);
        digitalWrite(in4, LOW);

    }

    if(receive == '1'){          //RIGHT ---
        digitalWrite(in1, HIGH);
        digitalWrite(in2, LOW);
        digitalWrite(in3, LOW);
        digitalWrite(in4, HIGH);

    }
  }
```

*Εικόνα 17 Συνάρτηση loop()*

## Προβλήματα & λύσεις

Κατά τη διαδικασία κατασκευής του project υπήρξαν ορισμένες δυσκολίες. Στις περισσότερες από αυτές βρέθηκαν λύσεις, όμως υπάρχουν και σημεία τα οποία επιδέχονται βελτίωση. Θα εξεταστούν αναλυτικά τα προβλήματα σε τρείς κατηγορίες.

### Προβλήματα python

Στον κώδικα της python δημιουργήθηκαν τα εξής προβλήματα:

- Infinity loop στη while ()

Λύση : Χρησιμοποιήθηκε η συνάρτηση waitKey() παίρνει σαν όρισμα το 1 ώστε να ενεργοποιηθεί στο αμέσως επόμενο πάτημα πλήκτρου Και να τερματίσει η while () επειδή όταν έκλεινε το πρόγραμμα δέσμευε τους πόρους της κάμερας και ο υπολογιστής δεν κατάφερνε να ξανα ανοίξει την κάμερα εάν δεν γινόταν επανακίνηση.

- Failed image processing

Ο αλγόριθμος επεξεργασίας εικόνας της Cv2 χρησιμοποιεί διαφορετική χρωματική διάταξη για την εμφάνιση εικόνας στο frame window (Blue Green Red) με αποτέλεσμα όλα να είναι μπλε και να μην καταφέρνει να επεξεργαστεί το χέρι για να αποδώσει τα Landmarks, έτσι και τοποθετήθηκε συνάρτηση αλλαγής χρωμάτων για την σωστή λειτουργία του αλγόριθμου.

- Run time processing

Το Bluetooth και το πρόγραμμα του Arduino ήταν σχετικά πολύ αργά σε σύγκριση με το Script της Python όχι όλες τις φορές αλλά αρκετές άνοιγε το image show frame γινόταν η επεξεργασία εικόνας στελνόταν δεδομένα και το ρομποτικό όχημα ανταποκρινόταν ύστερα από 2 δευτερόλεπτα. Οπότε από δοκιμές και διορθώσεις κατέληξε να είναι 2 δευτερόλεπτα η τελική καθυστέρηση που ορίστηκε.

- Παράθυρο img_show παραμένει ανοιχτό μετά το κλείσιμο του προγράμματος

Ίσως και να φταίει το λογισμικό που έτρεχε το script ( Pycharm ) αλλά σαν λύση ύστερα από έρευνα στο διαδίκτυο βρέθηκε η destroyAllWindows του cvzone που δημιουργήθηκε για αυτούς τους λόγους και κλείνει όλα τα παράθυρα όταν τερματίζεται το πρόγραμμα.

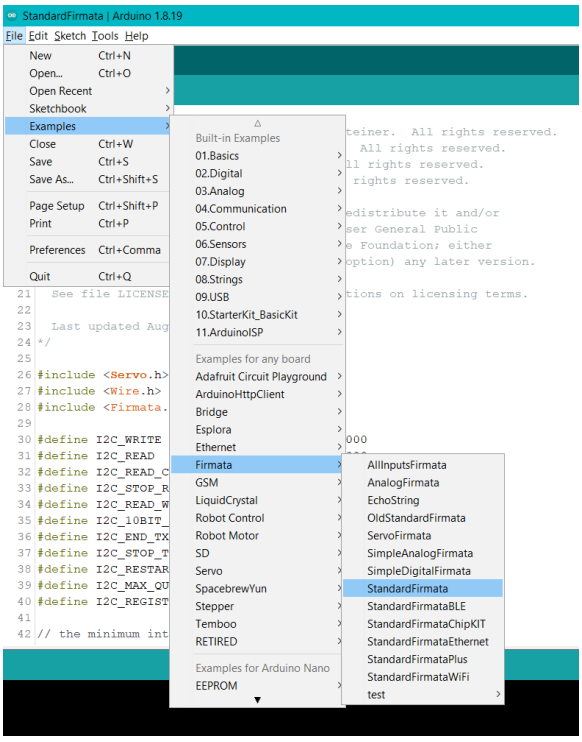
## Προβλήματα arduino

Στο κομμάτι του arduino υπήρξαν δύο προβλήματα: το ένα αφορούσε το προγραμματιστικό κομμάτι, ενώ το δεύτερο την συνδεσμολογία.

Στις πρώτες προσπάθειες για την υλοποίηση του remote controlled car χρησιμοποιήθηκαν λίγο διαφορετικοί κώδικες από αυτούς που αναφέρθηκαν στις προηγούμενες ενότητες. Ο κώδικας της python ήταν ο βασικός κώδικας, αποτελούμενος και πάλι από δύο script *main* και *controller*, όμως το περιεχόμενο του *controller* ήταν διαφορετικό. Χρησιμοποιήθηκε η βιβλιοθήκη **pyFirmata** η οποία αρχίζει μια σειριακή επικοινωνία με το arduino μέσω μιας πόρτας COMX την οποία την ορίζει ο προγραμματιστής. Για να γίνει η σύνδεση με το arduino, το sketch που φορτώνεται στο αναπτυξιακό είναι ένα παράδειγμα που περιέχεται στη βιβλιοθήκη *Firmata*, το *StandardFirmata*. Επειδή ακριβώς ο κώδικας για το arduino είναι έτοιμος, ο έλεγχος των κινητήρων γινόταν από το script *controller*.

```python
import pyfirmata
comport='COM6'
board=pyfirmata.Arduino(comport)
motor_1=board.get_pin('d:13:o')
motor_2=board.get_pin('d:12:o')
motor_3=board.get_pin('d:11:o')
motor_4=board.get_pin('d:10:o')
def motor(total):
    if total==0:    #stop moving
        motor_1.write(0)
        motor_2.write(0) #
        motor_3.write(0) #
        motor_4.write(0) #
    elif total==1:  #go right
        motor_1.write(1)
        motor_2.write(0)
        motor_3.write(1)
        motor_4.write(0)
    elif total==2:  #go left
        motor_1.write(0)
        motor_2.write(1)
        motor_3.write(0)
        motor_4.write(1)
    elif total==3:  #stop moving
        motor_1.write(0)
        motor_2.write(0)
        motor_3.write(0)
        motor_4.write(0)
    elif total==4:  #stop moving
        motor_1.write(0)
        motor_2.write(0)
        motor_3.write(0)
        motor_4.write(0)
    elif total==5:  #go forward
        motor_1.write(1)
        motor_2.write(1)
        motor_3.write(1)
        motor_4.write(1)
```

*Εικόνα 18 Παλιός κώδικας script controller*




*Εικόνα 19 StandardFirmata*

Το πρόβλημα υπήρξε στην επικοινωνία μέσω bluetooth με το HC05 module διότι δεν συγχρόνιζε, λόγω κάποιας αστοχίας είτε στο κομμάτι του κώδικα είτε στην κατάλληλη επιλογή σειριακής πόρτας. Αντιμετωπίστηκε με τους νέους κώδικες και την χρήση της βιβλιοθήκης Serial.

Όσο αν αφορά το δεύτερο πρόβλημα, χρειάζεται να αναφερθεί ότι αρχικά η υλοποίηση του κυκλώματος θα γινόταν με το Arduino UNO και όχι το NANO. Η λογική ήταν να χρησιμοποιηθεί το **L293D motor shield** που είναι πλήρως συμβατό με το UNO και οι κινητήρες να ήταν τέσσερις,

δηλαδή να υπήρχε κίνηση σε όλους τους τροχούς. Όμως τα digital pins του arduino UNO καλύπτονταν από το shield και δεν υπήρχε τρόπος να συνδεθεί το HC05.

*Εικόνα 20 L293D Motor shield*

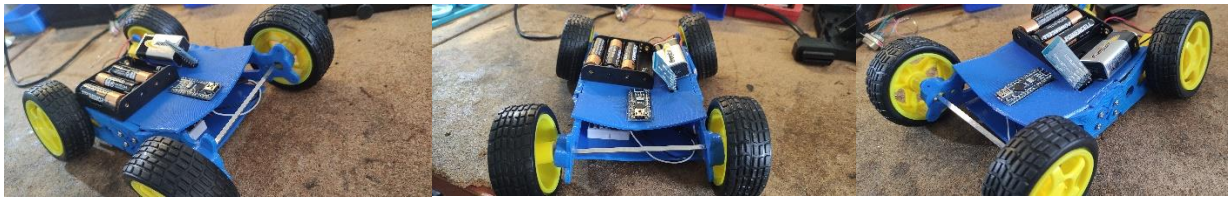
Επιπλέον στο L293D Motor shield που προοριζόταν για χρήση, το ένα από τα τρία ολοκληρωμένα ήταν κατεστραμμένο, οπότε θα καθιστούσε αδύνατο τον έλεγχο κινητήρων.

Ως λύση στο πρόβλημα, αφαιρέθηκε το λειτουργικό ολοκληρωμένο από το shield και χρησιμοποιήθηκε μονάχα αυτό, κάτι όμως που κατέστησε την χρήση τεσσάρων κινητήρων αδύνατη. Έτσι το τελικό όχημα έχει κίνηση μόνο στους πίσω τροχούς.

## Κατασκευαστικά προβλήματα

Όπως αναφέρθηκε και προηγουμένως, υπήρξαν προβλήματα στην κατασκευή του σκελετού του οχήματος. Το PLA δεν είναι ανθεκτικό στην ζέστη, έτσι όταν το όχημα εκτέθηκε σε υψηλή θερμοκρασία περιβάλλοντος (στο εσωτερικό της καμπίνας ενός αυτοκινήτου), παραμορφώθηκε. Η παραμόρφωση αυτή κατέστησε το όχημα ανίκανο να κινηθεί σωστά, έτσι έπρεπε να δημιουργηθεί νέος σκελετός.

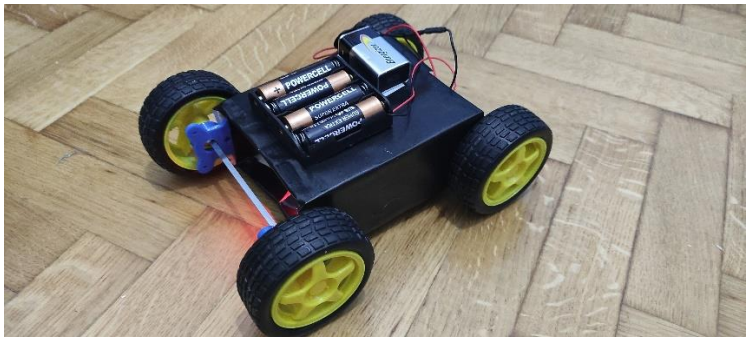
Στο νέο σκελετό διατηρήθηκαν δυο από τα κομμάτια του παλιού διότι δεν ήταν δυνατή η αφαίρεσή τους και το νέο περίβλημα δημιουργήθηκε από φύλλα ξύλου μπάλσας, ώστε να μην υπάρξει ξανά πρόβλημα με μεγάλες θερμοκρασίες.



*Εικόνα 21 Παραμορφωμένος σκελετός*

## ΑΝΑΣΚΟΠΗΣΗ – ΤΕΛΙΚΟ ΑΠΟΤΕΛΕΣΜΑ

Το τελικό αποτέλεσμα του Remote Controlled Car με Computer Vision είναι μία πλήρως λειτουργική διάταξη. Χειριζόμενο από την κάμερα του υπολογιστή το όχημα μπορεί και κινείται ελευθέρα στο χώρο, μέσω bluetooth πρωτοκόλλου.

*Εικόνα 22 Τελική κατασκευή*

Το κόστος ανήλθε στα 30€ και ο χρόνος υλοποίησης ήταν 3 εβδομάδες.

## ΒΕΛΤΙΩΣΕΙΣ

Το τελικό project είναι πλήρως λειτουργικό, όμως επιδέχεται ορισμένες βελτιώσεις σε διάφορους τομείς. Οι κυριότεροι είναι ο τομέας ελέγχου, ο κατασκευαστικός τομέας και ο προγραμματιστικός τομέας.

Αναλυτικότερα στον τομέα ελέγχου θα ήταν θετικό να υπήρχε η δυνατότητα να ελεγχθεί η κίνηση του οχήματος μέσω χειρονομιών και όχι απλά με το ανεβοκατέβασμα των δακτύλων. Έτσι θα ήταν ευκολότερο στον χρήστη να χειριστεί το όχημα. Ακόμα μια βελτίωση θα μπορούσε να είναι η δυνατότητα αναγνώρισης περισσότερων του ενός χεριού, διότι στην παρούσα φάση αναγνωρίζεται μόνο ένα χέρι τη φορά. Επιπλέον το αριστερό χέρι απεικονίζεται σαν καθρέφτης του δεξιού, κάτι το οποίο ενδέχεται να μπερδέψει τον χρήστη. Επίσης η προσθήκη αισθητήρων και καμερών πάνω στο όχημα θα ήταν μια αναβάθμιση, καθώς θα μπορούσε να επιστρέφει πληροφορίες από το περιβάλλον του.

Στον κατασκευαστικό τομέα καλό θα ήταν να επιλεγεί μια πιο στιβαρή κατασκευή από τις δύο που εφαρμόστηκαν, καθώς το PLA παραμορφώνεται σε μεγάλες θερμοκρασίες ενώ τα φύλλα ξύλου μπάλσας δεν είναι ιδιαίτερα στιβαρά. Μια λύση θα ήταν πάλι η εκτύπωση κομματιών από 3D εκτυπωτή, όμως το υλικό εκτύπωσης να ήταν διαφορετικό.

Τέλος, στο κομμάτι του προγραμματισμού οι δυνατότητες βελτίωσης είναι πολλές καθώς εξαρτάται από την ικανότητα του προγραμματιστή να φτιάξει ένα πιο γρήγορο και ελαφρύ πρόγραμμα. Μια χρήσιμη προσθήκη θα ήταν οι κατάλληλες εντολές για να είναι εφικτή η οπισθοδρόμηση του οχήματος. Επίσης καλό θα ήταν να αντιμετωπιστεί ένα πρόβλημα στο οποίο δεν βρέθηκε λύση: με το κλείσιμο του προγράμματος να σταματάει και η λειτουργία των κινητήρων, διότι τώρα συνεχίζουν να διατηρούν την προηγούμενή τους κατάσταση.

# ΜΕΛΛΟΝΤΙΚΕΣ ΕΦΑΡΜΟΓΕΣ

Ο τηλεχειρισμός μέσω κάμερας (computer vision) δύναται να έχει ποικίλες εφαρμογές. Το παρόν project με κάποιες τροποποιήσεις είναι δυνατόν να χρησιμοποιηθεί σε διάφορες εφαρμογές. Θα μπορούσε από ρομπότ ψυχαγωγίας να γίνει ρομπότ συλλογής δεδομένων περιβάλλοντος ή με μια άλλη διάταξη να γίνει ένα αντίγραφο του χεριού το οποίο θα κουνιέται ανάλογα με τις κινήσεις του χειριστή. Θα μπορούσε να εξελιχθεί μέχρι και σε ρομπότ παροχής βοήθειας αν θεωρηθεί ότι ο χειριστής είναι κωφάλαλος, και έχει προγραμματιστεί κατάλληλα ώστε να καταλαβαίνει την νοηματική γλώσσα.

# ΔΙΚΤΥΟΓΡΑΦΙΑ


Πληροφορίες για την ρομποτική και τα ρομπότ ανακτήθηκαν από

- https ://el.wikipedia.org/wiki/Ρομποτική Μάιος 2022
- http://users.sch.gr/jenyk/index.php/robotics Μάιος 2022

Πληροφορίες για το computer vision ανακτήθηκαν από

- https://www.geeksforgeeks.org/opencv-python-tutorial/ Μάιος 2022
- www.roboticstomorrow.com Μάιος 2022

Πληροφορίες για το Mediapipe ανακτήθηκαν από

- https://google.github.io/mediapipe/solutions/hands#python-solution-api Μάιος 2022

Πληροφορίες για το script controller ανακτήθηκαν από

- https://www.programcreek.com/python/example/92299/controller.Controller Μάιος 2022
- https://pypi.org/project/controller/ Μάιος 2022

Ενδεικτικός κώδικας για το arduino και πληροφορίες για το L293D ανακτήθηκαν από

- https://lastminuteengineers.com/l293d-dc-motor-arduino-tutorial/ Μάιος 2022

Πληροφορίες για το Arduino ανακτήθηκαν από

- https://flaviocopes.com/arduino-programming-language/ Ιούνιος 2022
- https://docs.arduino.cc/static/4fa69364e034ae0c9d965408b760136c/A000005-datasheet.pdf Ιούνιος 2022

Πληροφορίες για το HC05 ανακτήθηκαν από

- https://components101.com/sites/default/files/component_datasheet/HC-05%20Datasheet.pdf Ιούνιος 2022

Τα σχέδια για την 3D εκτύπωση ανακτήθηκαν από

- https://www.thingiverse.com/thing:4575879/files Μάιος 2022

Πληροφορίες για την βιβλιοθήκη SoftwareSerial ανακτήθηκαν από

- https://docs.arduino.cc/learn/built-in-libraries/software-serial Ιούνιος 2022