Ατομική Διπλωματική Εργασία

ΕΝΕΡΓΕΙΑΚΟ ΠΡΟΦΙΛ ΕΦΑΡΜΟΓΩΝ ΡΟΜΠΟΤΙΚΟΥ ΒΡΑΧΙΟΝΑ & ΕΞΥΠΝΗΣ ΚΑΜΕΡΑΣ

Ρεβέκκα Θεοδώρου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2024

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ενεργειακό Προφίλ Εφαρμογών Ρομποτικού Βραχίονα & Έξυπνης Κάμερας

Ρεβέκκα Θεοδώρου

Επιβλέπων Καθηγητής
Δρ. Μάριος Δ. Δικαιάκος

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή μου, Δρ. Μάριο Δ. Δικαιάκο για την ευκαιρία να δουλέψω σε ένα τόσο ενδιαφέρον θέμα, και την καθοδήγηση του κατά τη διάρκεια εκπόνησης της διπλωματικής μου εργασίας.

Θα ήθελα επίσης να ευχαριστήσω το Κέντρο Επιχειρηματικότητας του Πανεπιστημίου Κύπρου για την δυνατότητα εργασίας στο εργαστήριο MAKERSPACE και την παροχή του εξοπλισμού για δημιουργία αυτής της εργασίας.

Περίληψη

Η τεχνητή νοημοσύνη αποτελεί ένα από τα πιο κύρια και απαραίτητα εφόδια του σημερινού ανθρώπου ως προς την επίλυση υφιστάμενων προβλημάτων σε διάφορους τομείς της ζωής του, όπως στον τομέα της υγείας, της εκπαίδευσης, της βιομηχανίας, αλλά και δημιουργίας νέων δυνατοτήτων και τεχνολογιών που να φέρουν την ανθρωπότητα ένα βήμα πιο κοντά στο μέλλον.

Στην παρούσα διπλωματική εργασία, παρατίθενται οι ενέργειες και τα πειράματα μου, ως προς την διαπίστωση των δυνατοτήτων της συνεργασίας ενός ρομποτικού βραχίονα και μίας έξυπνης κάμερας. Στόχευσα στην δημιουργία προσομείωσης ενός βιομηχανικού σεναρίου το οποίο συνδιάζει τη ρομποτική με την τεχνητή νοημοσύνη και τη μηχανική μάθηση. Τέλος, ερεύνησα την ενεργειακή κατανάλωση του ρομποτικού βραχίονα και της έξυπνης κάμερας καθώς και το ποσοστό επιτυχίας της εκτέλεσης μιας λειτουργίας από τα δύο αυτά εξαρτήματα. Εργάστηκα χρησιμοποιώντας τις δυνατότητες και τα μέσα που είναι διαθέσιμα στο εργαστήριο MAKERSPACE του Κέντρου Επιχειρηματικότητας του Πανεπιστημίου Κύπρου. Εξήγαγα διαπιστώσεις που θα μπορούσαν να χρησιμοποιηθούν απο μελλοντικούς χρήστες των εν λόγω μηχανημάτων. Προέκυψαν όμως στην πορεία και κάποια προβλήματα, των οποίων παραθέτω τη λύση.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Ρομποτική και Τεχνητή Νοημοσύνη

Στην αρχή του 20ου αιώνα, η επιστημονική φαντασία μύησε στον κόσμο την έννοια των τεχνητά ευφυών ρομπότ. Μέχρι τη δεκαετία του 1950 μια ομάδα επιστημόνων, μαθηματικών και φιλοσόφων άρχισε να ασχολείται με την έννοια της τεχνητής νοημοσύνης. Ένας από αυτούς τους επιστήμονες ήταν και ο Άλαν Τούρινγκ. [4]

Τα τελευταία χρόνια, η τεχνητή νοημοσύνη έχει εξελιχθεί σημαντικά και έχει εισβάλει σε πολλούς άλλους τομείς της επιστήμης, προσφέροντας τους νέες ιδέες και δυνατότητες. Ένας τέτοιος τομέας είναι η ρομποτική. Ο συνδιασμός της ρομποτικής με την τεχνητή νοημοσύνη έχει αυξήσει την παραγωγικότητα, αποδοτικότητα, ευελιξία και ασφάλεια για τους ανθρώπους. [5]

Ένας από τους σημαντικότερους κλάδους της τεχνητής νοημοσύνης αποτελεί η μηχανική μάθηση. Η μηχανική μάθηση είναι ο κλάδος της τεχνητής νοημοσύνης που στοχεύει στην εκμάθηση συστημάτων μέσω δεδομένων. Επιτρέπει συγκεκριμένα, σε ένα υπολογιστή να εντοπίζει μοτίβα συμπεριφοράς ανάμεσα στα δεδομένα και να προσπαθεί να προβλέψει ένα επικείμενο αποτέλεσμα.

Στον τομέα της ρομποτικής η συμβολή της μηχανικής μάθησης αφορά την εκμάθηση εργασιών στα ρομπότ μέσω παρατήρησης και μίμησης ανθρωπίνων ενεργειών. Ένα τέτοιο ρομπότ μπορεί να παρατηρήσει το περιβάλλον του και να συλλέξει δεδομένα μέσω αισθητήρων τα οποία έπειτα θα επεξεργαστεί και θα μετατρέψει σε γνώση που θα εφαρμόσει στην πορεία. Υπάρχει αυτονομία στην εκμάθηση των ρομπότ λόγω των αλγορίθμων

μηχανικής μάθησης. Τα ρομπότ που αξιοποιούν τη μηχανική μάθηση χρησιμοποιούνται σε μια πληθώρα εφαρμογών όπως την επιθεώρηση, συντήρηση, επιτήρηση, μεταποίηση, ακόμα και στην υγειονομική περίθαλψη. Στην βιομηχανία, τα ρομπότ αυτά, μπορούν να βελτειώσουν σημαντικά την ακρίβεια, ταχύτητα και αποδοτικότητα των διαδικασιών [6].

Για να μειώσουμε την καθυστέρηση του δικτύου (latency) στην εκτέλεση εντολών σε ρομποτικές εφαρμογές, χρησιμοποιούμε το edge computing υπό μορφή υπολογιστικών κόμβων τους οποίους τοποθετούμε κοντά στα εξαρτήματα του συστήματος. Στην περίπτωση της εφαρμογής του σε εφαρμογές τεχνητής νοημοσύνης στη ρομποτική, το edge computing ανταποκρίνεται στην ερμηνεία μεγάλων ποσοτήτων δεδομένων. Οι αισθητήρες των ρομπότ συλλέγουν τεράστια ποσότητα δεδομένων σε πραγματικό χρόνο, τα οποία πρέπει να επεξεργαστούν. Με τη βοήθεια του edge computing, αντί να στέλνονται στο cloud για επεξεργασία, τα δεδομένα αναλύονται στη μηχανή. Η ταχύτητα αυξάνεται και ο χρόνος εκτέλεσης της επεξεργασίας μειώνεται σημαντικά. [7][8]

Η κατανάλωση ενέργειας είναι ένας από τους σημαντικούς παράγοντες που λαμβάνουμε υπόψη κατά τον έλεγχο απόδοσης ενός βιομηχανικού ρομπότ. Η βελτιστοποίηση της μπορεί να μειώσει σημαντικά το κόστος λειτουργίας σε χρόνο και πόρους. Με την εφαρμογή συστημάτων διαχείρησης της ενέργειας, η κατανάλωση ενέργειας ενός ρομπότ μπορεί να παρακολουθείται ώστε να ερευνηθεί η βελτιστοποίηση της κατά την εκτέλεση μιας εργασίας [15] .

1.2 Σκοπός εργασίας

Σκοπός της εργασίας μου είναι να συνδιάσω την ρομποτική με την τεχνητή νοημοσύνη και την μηχανική μάθηση για να βελτιώσω την εκτέλεση ρομποτικών λειτουργιών και να παρατηρήσω τα θετικά και αρνητικά που προσφέρει. Συνοπτικά, να κάνω την ενεργειακή σκιαγράφηση των λειτουργιών αυτών. Εφάρμοσα το πείραμα μου σε ένα χώρο που αποτελεί προσομοίωση ενός βιομηχανικού σεναρίου με ρομποτικές μηχανές. Σε ένα σενάριο pick and place αντικειμένων με βάση κάποιο χαρακτηριστικό τους, από ένα ρομποτικό βραχίονα.

Στόχος λοιπόν της Διπλωματικής μου εργασίας είναι η εκτέλεση πειραμάτων που σχετίζονται με την κατανάλωση ενέργειας κατά τη λειτουργία μιας εφαρμογής η οποία συνδιάζει τον ρομποτικό βραχίονα, τον ιμάντα και την έξυπνη κάμερα. Η λειτουργία αυτή αποτελεί την

συλλογή και τοποθέτηση κύβων από τον ιμάντα σε άλλο σημείο του χώρου, με βάση το χρώμα τους. Την συλλογή εκτελεί ο ρομποτικός βραχίονας σε συνεργασία με την έξυπνη κάμερα. Για σκοπούς δημιουργίας ενός ενεργειακού προφίλ της λειτουργίας, έχω εκτελέσει πειράματα ενεργειακής κατανάλωσης με διάφορες μεταβλητές της λειτουργίας.

1.3 Δομή εργασίας

Στο πρώτο κεφάλαιο της εργασίας μου, γίνεται μια αναφορά στους στόχους και θέμα της εργασίας. Παρουσιάζεται σε συντομία το υπόβαθρο του τομέα με τον οποίο ασχολείται αυτή η εργασία και ο σκοπός υλοποίησης της.

Στο δεύτερο κεφάλαιο, παρουσιάζεται η αρχιτεκτονική του συστήματος καθώς και τα πιο σημαντικά στοιχεία του. Το υλικό (ο ρομποτικός εξοπλισμός) που χρησιμοποιήθηκε καθώς και τα απαραίτητα λογισμικά για διαχείριση του. Συγκεκριμένα, το 2ο υποκεφάλαιο αφορά τον ρομποτικό βραχίονα και ρομποτικό ιμάντα Dobot Magician τα οποία αποτελούν και τον ρομποτικό εξοπλισμό του συστήματος. Το 3ο υποκεφάλαιο είναι μια εισαγωγή στο λογισμικό του Dobot Magician, DobotStudio και τις λεπτομέρειες λειτουργίας του. Το 4ο υποκεφάλαιο είναι αφιερωμένο στην έξυπνη κάμερα JeVois A33 η οποία αποτελεί την τεχνολογία τεχνητής νοημοσύνης-μηχανικής μάθησης του συστήματος. Τέλος το 5ο υποκεφάλαιο αφορά το λογισμικό για χειρισμό της κάμερας JeVois, JeVois Inventor.

Στο τρίτο κεφάλαιο της εργασίας παραθέτεται η λειτουργία που δημιουργήθηκε και συνδιάζει τη λειτουργία όλων των εξαρτημάτων μαζί. Παρουσιάζεται η διάταξη του συτήματος στο 1ο υποκεφάλαιο και έπειτα κάθε υποκεφάλαιο που ακολουθεί αφορά μία λειτουργία.

Το τέταρτο κεφάλαιο αφορά στην εκτέλεση πειραμάτων στο σύστημα. Αναλύει αρχικά τη μεθοδολογία, την πειραματική διάταξη και τις πειραματικές μεταβλητές που συντάσσουν το πείραμα. Έπειτα, παρατηρεί τα αποτελέσματα των πειραμάτων για κάθε λειτουργία και τους πειραματικούς πόρους. Τα πειράματα αυτά στοχεύουν στην παρατήρηση της αλλαγής των αποτελεσμάτων εαν τροποποιηθούν κάποιες μεταβλητές της λειτουργίας, καθώς και τις μετρήσεις κατανάλωσης ενέργειας κατά τη διάρκεια των αλλαγών.

Το πέμπτο και τελευταίο κεφάλαιο αφορά τα συμπεράσματα που παράχθηκαν από τα προηγούμενα βήματα και τη μελλοντική εργασία που μπορεί να προκύψει.

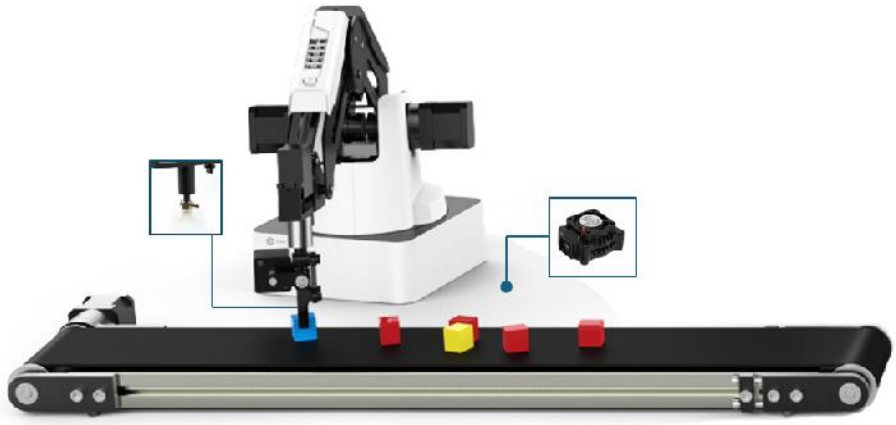
Τέλος, παρατείθονται η βιβλιογραφία και τα παραρτήματα του κώδικα που δημιούργησα για τη λειτουργία τους συστήματος.

Κεφάλαιο 2

Περιγραφή Συστημάτων

2.1 Διάταξη συστήματος

Το σύστημα αποτελείται από τον Ρομποτικό Βραχίονα Dobot Magician, τον κυλιώμενο ιμάντα του Dobot Magician και την JeVois A33 Smart Camera.
Ο ρομποτικός βραχίονας συνδέεται μέσω θύρας USB με τον υπολογιστή από όπου ο χρήστης του συστήματος μπορεί να τον χειρίζεται. Η JeVois κάμερα ενώνεται επίσης μέσω θύρας USB με τον υπολογιστή, του οποίου στέλνει τα σήματα που λαμβάνει από το περιβάλλον του συστήματος.

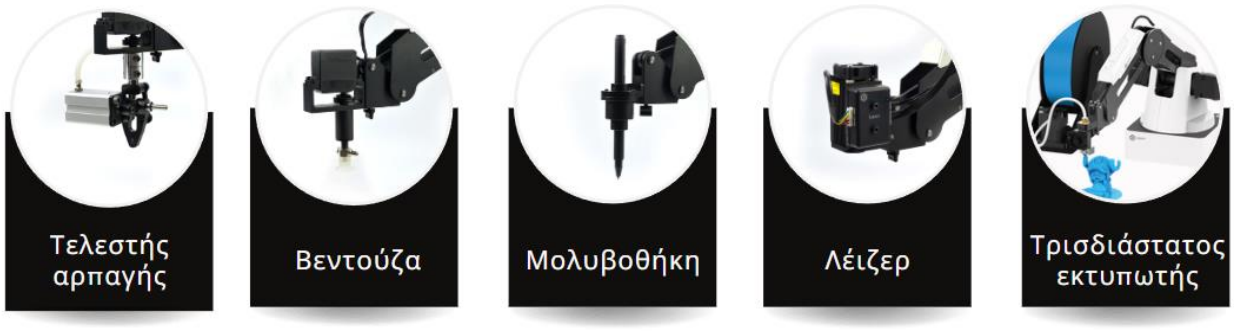


Εικόνα 1: Πειραματική διάταξη εξοπλισμού

2.2 Dobot Magician Robot

Το Magician Robot της εταιρίας Dobot είναι το πρώτο εκπαιδευτικό επιτραπέζιο ρομπότ με κίνηση και στους 4 άξονες, στον κόσμο. Παρέχει την δυνατότητα ανάπτυξης ποικίλων εφαρμογών μέσω της χρήσης λογισμικού και αξεσουάρ υλικού και υποστηρίζει δευτερογενή ανάπτυξη με 13 επεκτάσιμες διεπαφές και πάνω από 20 γλώσσες προγραμματισμού.
Σε αναγνώριση των εξαιρετικών επιδόσεων του στον σχεδιασμό του υλικού και την εφαρμογή λογισμικού, το DOBOT Magician έχει λάβει το βραβείο καινοτομίας CES 2018 και το iF DESIGN AWARD το 2018 [1].

Το Magician διαθέτει πληθώρα λειτουργιών όπως τρισδιάστατη εκτύπωση, χάραξη με λέιζερ, καλλιγραφία και σχέδιο, καθώς και μεγάλη ποικιλία αξεσουάρ όπως αισθητήρων και θυρών. Για την πραγματοποίηση των διαφόρων του λειτουργιών έχει πέντε (5) συμβατούς τελικούς τελεστές που παρουσιάζονται και στην εικόνα πιο κάτω: τον τελεστή αρπαγής, την βεντούζα, τη μολυβοθήκη, το λέιζερ και τον τρισδιάστατο εκτυπωτή.



Εικόνα 2 Dobot Magician end effectors

Λειτουργία του Dobot Magician:
Ο ρομποτικός βραχίονας Magician ενεργοποιείται μέσω του κουμπιού Power που βρίσκεται στην βάση του. Πατώντας το κουμπί λειτουργίας, ένα πορτοκαλί φως ανάβει. Για να ενεργοποιηθεί σωστά ο βραχίονας και το φως να γίνει πράσινο, τοποθετούμε τον κορμό του βραχίονα στην «ουδέτερη θέση» σχηματίζοντας γωνιά 45 μοιρών μεταξύ αντιβραχίου και πίσω βραχίονα. Ο κορμός του βραχίονα μπορεί να μετακινηθεί μόνο εαν ταυτόχρονα με την κίνηση κρατάμε πατημένο του κουμπί Lock.

Πέντε δευτερόλεπτα μετά την τοποθέτηση του βραχίονα στην «ουδέτερη θέση» ακούγεται ένας ήχος και ανάβει το πράσινο φωτάκι. Τότε μπορούμε να χειριστούμε ελεύθερα τον βραχίονα. Εαν το φως γίνει κόκκινο υποδυκνύει ότι ο βραχίονας είναι σε περιορισμένη θέση και πρέπει να επανατοποθετηθεί σωστά.

Για απενεργοποίηση του βραχίονα πατάμε ξανά το κουμπί λειτουργίας (Power). Έπειτα από την δράση αυτή, το Dobot θα μετακινηθεί στην θέση «ανάπαυσης». Πρέπει να προσέξουμε να μην υπάρχουν αντικείμενα στον χώρο αυτό ώστε να μην δημιουργηθούν ατυχήματα, και ο βραχίονας να τερματίσει σωστά.

Σε περίπτωση που ο βραχίονας δεν είναι ευθυγραμμισμένος με τον υπολογιστή, πατάμε το κουμπί επαναρύθμισης (Reset). Το Dobot θα αποσυνδεθεί από τον υπολογιστή και θα επαναρυθμιστεί.

Για σύνδεση περιφερειακών συσκευών στον βραχίονα πρέπει να απενεργοποιήσουμε τον βραχίονα, για να αποφύγουμε ζημιές στο Dobot.

Το Dobot Magician συνδέεται με τον υπολογιστή μέσω καλωδίου USB.

Χαρακτηριστικά Ρομποτικού Βραχίονα:

- Αριθμός αξόνων: 4
- Ωφέλιμο φορτίο: 500 g
- Μέγιστη Εμβέλεια: 320 mm
- Επαναληψημότητα: ±0,2 mm
- Τροφοδοσία 100V - 240V, 50/60 HZ
- Τροφοδοσία 12V / 6,5A DC
- Κατανάλωση: 78W Max

Χαρακτηριστικά ιμάντα:

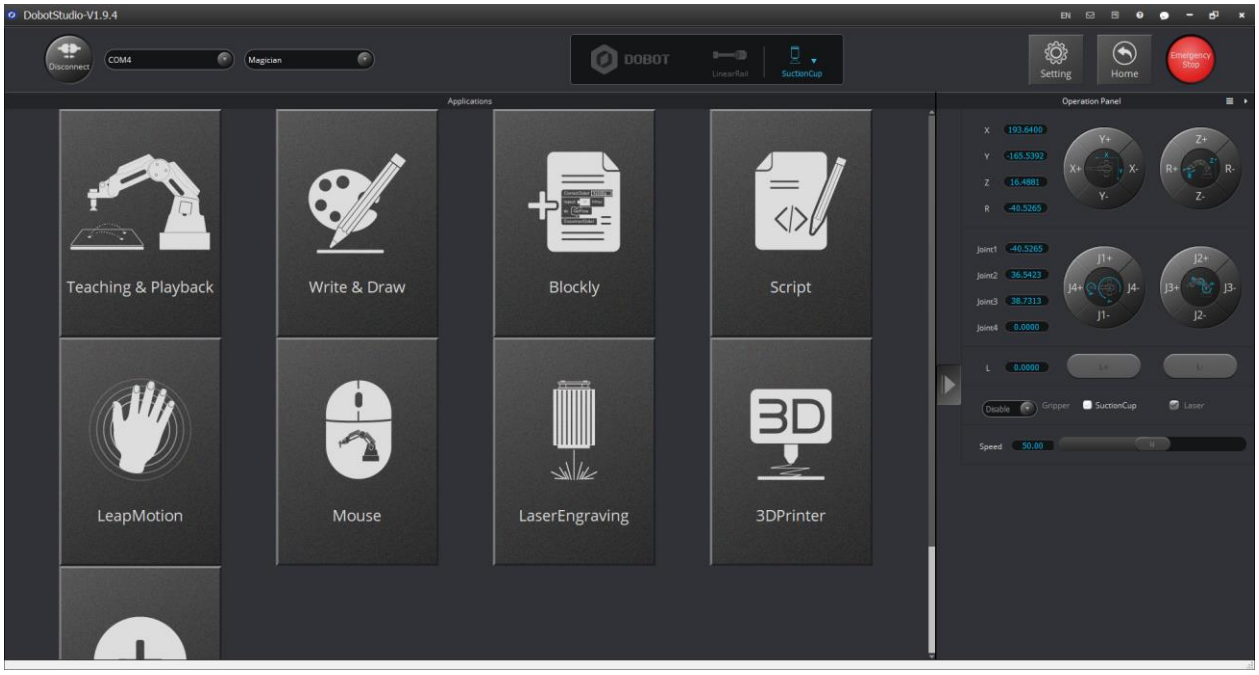
- Ωφέλιμο φορτίο: 500 g
- Αποτελεσματική απόσταση παράδοσης: 600 mm
- Μέγιστη ταχύτητα: 120 mm/s
- Μέγιστη επιτάχυνση: 1100 mm/s^2

2.3 Dobot Studio

Για διαχείρηση του ρομποτικού βραχίονα και των εξαρτημάτων του, καθώς και για τον προγραμματισμό και την δοκιμή λειτουργιών σ'αυτό, χρησιμοποιήθηκε το λογισμικό DobotStudio.

Το DobotStudio είναι το επίσημο λογισμικό ελέγχου ρομπότ της Dobot. Υποστηρίζει τα Magician, Magic Box και Magician Lite. Υποστηρίζει επίσης λειτουργίες όπως το Blockly, Script, Teaching and Payback και Write and Draw.

Για την υλοποίηση της παρούσας εργασίας χρησιμοποίησα το DobotStudio (DOBOT Magician), version 1.9.4 που δημοσιεύθηκε στις 10 Ιανουαρίου, το 2022.



Εικόνα 3 DobotMagician Αρχική Σελίδα

Το DobotStudio δίνει στους χρήστες τους δύο διαφορετικές επιλογές προγραμματισμού του Magician. Την προγραμματιστική γλώσσα Blockly και το Script, που αποτελεί την υψηλού επιπέδου γλώσσα προγραμματισμού Python.

Η Blockly [2] είναι μια γλώσσα προγραμματισμού με την μορφή οπτικών μπλοκ (block) κώδικα που θυμίζουν κομμάτια παζλ. Τα μπλοκ συνδέονται μεταξύ τους και δημιουργούν μια ακολουθία εντολών (κώδικα) ο οποίος έπειτα μεταφράζεται σε Python για εκτέλεση. Είναι μια πολύ απλή στην κατανόηση γλώσσα που επιτρέπει στους χρήστες της να προγραμματίζουν με ένα πιο δημιουργικό τρόπο, χωρίς να ανησυχούν για τη σύνταξη των εντολών.
Η Python είναι μια διερμηνευτική, αντικειμενοστρεφής γλώσσα με δυναμική σημασιολογία. Είναι εύκολη στην εκμάθηση και απλή στη συγγραφή. Είναι η ιδανική γλώσσα προγραμματισμού ρομπότ σαν κι αυτό λόγω του ότι δεν χρειάζεται ξεχωριστό μεταγλωττιστή.

Πέραν της δυνατότητας προγραμματισμού του Dobot Magician μέσω του Blockly και Script, το DobotStudio παρέχει τις εφαρμογές Teaching & Playback, Write & Draw, LeapMotion, Mouse, LaserEngraving και 3DPrinter.

- Το Teaching & Playback (Διδασκαλία & Αναπαραγωγή) είναι σύστημα για εκμάθηση κίνησης στο Dobot. Επιτρέπει στο Dobot να εκτελέσει καταγεγραμμένες κινήσεις με χειροκίνητο έλεγχο.
- Το Write & Draw (Εγγραφή & Σχεδίαση) είναι σύστημα ελέγχου του Dobot για να γράψει, να σχεδιάσει ή να χαράξει με το λέιζερ.
- Το Leap Motion (Άλμα κίνησης) προσφέρει έλεγχο του Dobot με χειρονομίες.
- Το Mouse (Ποντίκι) προσφέρει έλεγχο του Dobot με την κίνηση του mouse.
- Το Laser Engraving (Χαρακτική λέιζερ) χαράζει εικόνες και κείμενο μέσω bitmap με το Dobot.
- Δίνεται επίσης η επιλογή Add More (Προσθήκη Περισσότερων) για προσθήκη περισσότερων λειτουργιών για το Dobot.

Μπορούμε να πραγματοποιήσουμε την σύνδεση μεταξύ Magician και DobotStudio μέσω του κουμπιού Connect στο πάνω αριστερό μέρος του λογισμικού.
Στα δεξιά της οθόνης του DobotStudio βρίσκονται τα χειριστήρια κουμπιά του βραχίονα. Με αυτά μπορούμε να μετακινήσουμε χειρονακτικά τον κορμό του βραχίονα καθώς και τον τελικό τελεστή του (end effector).

Dobot API:

Το Dobot Magician API εξασφαλίζει την συνδεσιμότητα μεταξύ Dobot Magician και περιφερειακών συσκευών του, όπως αισθητήρων και ιμάντα. Εντοπίζεται ως βιβλιοθήκη σε Blockly και Python.

Για την εργασία αυτή, χρησιμοποίησα το Dobot API με το ειδικό αρχείο υλοποίησης DobotDllType.py που προσφέρει η Dobot. Το αρχείο DobotDllType.py ενθυλακώνει τη διεπαφή τύπου C του Dobot DLL, η οποία είναι το Python API της Dobot.

Το Dobot API χρησιμοποιεί λειτουργία ουράς (queue mode) για κλήσεις που σχετίζονται με την κίνηση του βραχίονα ή του ιμάντα. Ακολουθούν οι εντολές του Dobot API που χρησιμοποίησα στην εργασία αυτή, με τις αντίστοιχες περιγραφές λειτουργίας τους.

Σημαντικό να αναφέρω ότι κατά την εισαγωγή της βιβλιοθήκης του Dobot API στο πρόγραμμα, χρησιμοποίησα την συντομογραφία «dType» για να αναφέρομαι σε αυτή.

Ένωση με Dobot:

- **dType.load()** : Φώρτωση DLL και λήψη αντικειμένου Store (API). Χρησιμοποιείται για κλήση του python API.
- **dType.ConnectDobot(api, "COM4", 115200)** : Σύνδεση με Dobot Magician.
- **dType.DisconnectDobot(api)** : Αποσύνδεση από Dobot Magician.

Ουράς:

- **dType.SetQueuedCmdClear(api)** : Εκκαθάριση της ουράς.
- **dType.SetQueuedCmdStartExec(api)** : Εκκίνηση εκτέλεσης ουράς.
- **dType.SetQueuedCmdStopExec(api)** : Τερματισμός εκτέλεσης ουράς.
- **dType.GetQueuedCmdCurrentIndex(api) :** Index εντολής που εκτελείται.

Ορισμός παραμέτρων κίνησης:

- **dType.SetHOMEParams(api, 200, 200, 200, 200, isQueued = 1)** : Ορισμός αρχικής θέσης.
- **dType.SetPTPJointParams(api, 200, 200, 200, 200, 200, 200, 200, 200, isQueued = 1)** : Ορισμός των παραμέτρων για PTP-Motion σε κοινό σύστημα συντεταγμένων.
- **dType.SetPTPCommonParams(api, 100, 100, isQueued = 1)** : Ορισμός των κοινών παραμέτρων του PTP-Motion.

- **dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode, xHome, yHome, zHome, rHead, isQueued = 1)** : Εκτέλεση PTP εντολής.

End effector:

- **dType.SetEndEffectorSuctionCup(api, 0, 0, isQueued = 1)** : Ορισμός κατάστασης του Suction Cup end effector.

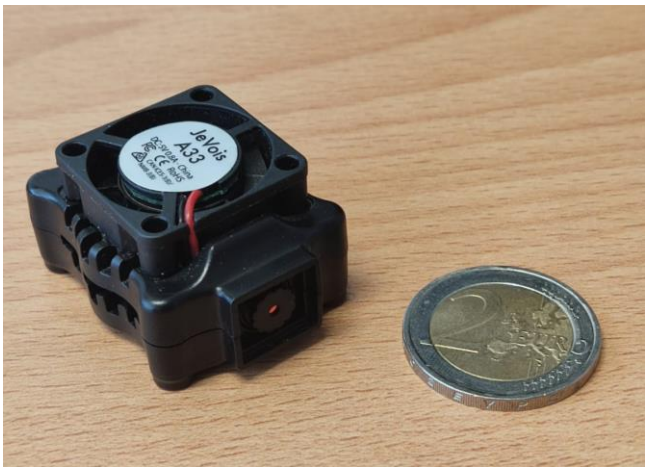
Ιμάντας μετακίνησης:

- **dType.SetEMotor(api, 0, 1, 10000, isQueued = 1)** : Ορισμός παραμέτρων ιμάντα, για κίνηση.

- **dType.dSleep(80)** : Ορισμός χρονικής καθυστέρησης. (Σε millisecond)

2.4 JeVois A33

Η JeVois A33 [14] είναι μια κάμερα επεξεργασίας της εταιρίας JeVois. Συνδιάζει ένα αισθητήρα βίντεο, ένα ενσωματωμένο τετραπύρηνο επεξεργαστή, σύνδεση βίντεο USB και σειριακή θύρα σε μόνο 28 κυβικά εκατοστά. Ο ισχυρός της επεξεργαστής 1,34GHz παρέχει αρκετή ισχύ για την εκτέλεση των 30 αλγορίθμων επεξεργασίας αναγνώρισης αντικειμένων, ανοιχτού κώδικα που προσφέρει η JeVois.



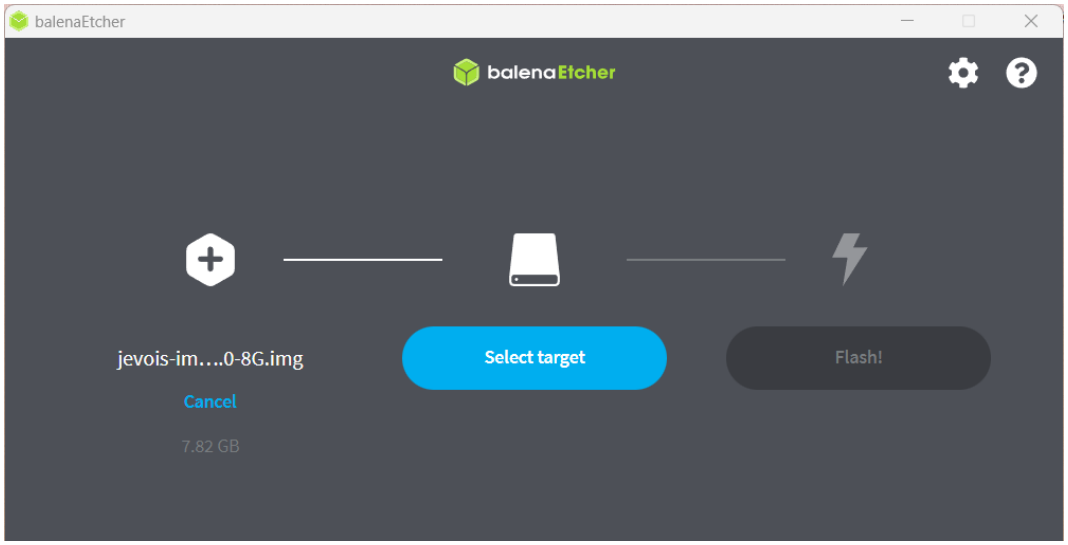
Εικόνα: 4 Η JeVois A33 κάμερα

Μέχρι στιγμής η JeVois κάμερα υποστηρίζει τους εξής αλγορίθμους, καθώς και πολλούς άλλους:

- Βαθιά νευρωνικά δίκτυα
- Ανίχνευση πλέγματος προσώπου
- Επιταχυντές DNN
- Εντοπισμός και αναγνώριση προσώπων ή αντικειμένων
- Παράδειγμα όρασης για το FIRST robotics 2018
- Ανίχνευση αντικειμένων που τραβούν την προσοχή
- Ανίχνευση δρόμων
- Ανίχνευση και αποκωδικοποίηση κωδικών QR και γραμμωτών κωδικών
- Ανίχνευση σήμανσης επαυξημένης πραγματικότητας
- Ανίχνευση και αποκωδικοποίηση ετικετών ArUno
- Παρακολούθηση αντικειμένων με βάση το χρώμα
- Υπολογισμός ροής κίνησης στα 100 Hz
- Φιλτράρισμα βίντεο με χρήση OpenGL-ES 2.0
- Τμηματοποίηση βίντεο σε υπερ-πίξελ
- Ανίχνευση ακμών σε 4 κλίμακες ταυτόχρονα
- Γρήγορο φιλτράρισμα εικόνας με χρήση NEON
- Εγγραφή βίντεο σε microSD
- Πυκνά σημεία κλειδιά SIFT
- Ανίχνευση αντικειμένων FIRST Robotics
- Παρακολούθηση ετικέτας ArUco και έγχρωμου μπλοκ

Αρχικοποίηση κάμερας:

Για χρήση της κάμερας JeVois πρέπει αρχικά να γίνει Flash η microSD κάρτα της. Αφαιρούμε την κάρτα SD από την JeVois και την ενώνουμε στον υπολογιστή μέσω κάποιου adapter. Κάνουμε Format την κάρτα και εγκαθιστούμε στον υπολογιστή μας την τελευταία έκδοση της JeVois A33 microSD εικόνας από την ιστοσελίδα της JeVois (8GB). Τότε, με τη χρήση του λογισμικού balenaEtcher κάνουμε Flash την εικόνα στην microSD κάρτα της JeVois και την τοποθετούμε ξανά πίσω στην κάμερα.



Εικόνα 5: Διαδικασία Flash μέσω του balenaEtcher

Για να λειτουργήσουμε την κάμερα JeVois, την ενώνουμε με σύρμα USB mini to USB στον υπολογιστή μας και μέσω του λογισμικού JeVois Inventor ή οποιουδήποτε άλλου λογισμικού βίνετο/κάμερας, όπως για παράδειγμα το AMCap, μπορούμε να παρακολουθήσουμε την έξοδο της κάμερας (βίντεο). Καθώς η κάμερα JeVois έχει πληθώρα λειτουργιών (modules), μπορούμε επίσης να εφαρμόσουμε το module που επιθυμούμε.

2.5 JeVois Inventor

Για διαχείριση της JeVois κάμερας και παρακολούθηση της εξόδου της (εικόνας), χρησιμοποιήθηκε το επίσημο λογισμικό της JeVois, JeVois Inventor. Είναι μια γραφική διεπαφή χρήστη για την κάμερα JeVois A33. Κάνει εύκολο τον προγραμματισμό νέων σωληνώσεων (pipeline) υπολογιστικής όρασης που τρέχουν στην κάμερα, με τη χρήση Python και OpenCV.

Η OpenCV (Open Source Computer Vision Library)[11] είναι μια ανοικτού κώδικα βιβλιοθήκη λογισμικού για υπολογιστική όραση και μηχανική μάθηση. Περιέχει περισσότερους από 2500 βελτιστοποιημένους αλγορίθμους, οι οποίοι μπορούν να χρησιμοποιηθούν για τον εντοπισμό και αναγνώριση προσώπων και αντικειμένων, την παρακολούθηση κινήσεων μιας κάμερας, την παρακολούθηση κινούμενων αντικειμένων, την

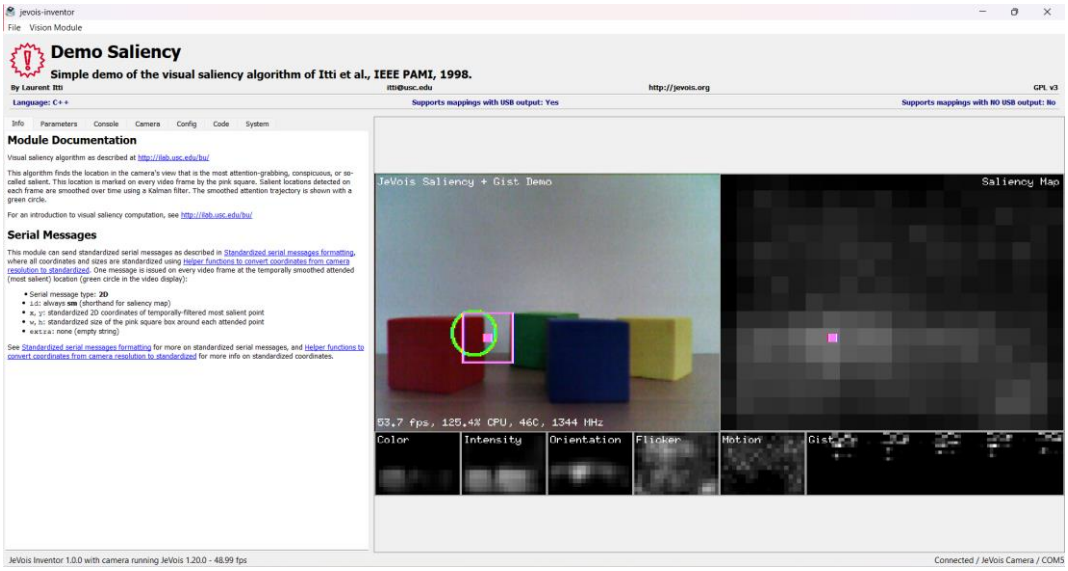
εξαγωγή τρισδιάστατων μοντέλων, την παραγωγή τρισδιάστατων νεφών (clouds) σημείων από στερεοφωνικές κάμερες, καθώς και πολλά άλλα. Είναι μια ευρέως γνωστη βιβλιοθήκη που αριθμεί πάνω από 47 χιλιάδες χρήστες και 18 εκατομμύρια λήψεις.

Εικόνα 6: Το λογότυπο της OpenCV

Το JeVois Inventor [12] εντοπίζει αυτόματα τα Modules από την κάρτα της κάμερας και δίνει τη δυνατότητα στους χρήστες του να επιλέξουν ποιό Module να χρησιμοποιήσουν ανα πάσα στιγμή. Είναι απλό στη χρήση και εύκολο στην κατανόηση. Πέραν από τη δυνατότητα εφαρμογής, δίνει στον χρήστη και την ευκαιρία να δημιουργήσει δικό του module στη γλώσσα Python.

Για εισαγωγή δεδομένων π.χ. εικόνων, στην κάρτα της κάμερας για να χρησιμοποιηθούν από κάποιο module, υπάρχει η επιλογή Enable - Export microSD card inside JeVois to host computer στο παράθυρο System του module στο JeVois Inventor. Πατώντας το κουμπί αυτό στο επιλεγμένο module, ανοίγει στη μορφή file η κάρτα microSD της JeVois και ο χρήστης μπορεί να επεξεργαστεί τα περιεχόμενα της.

Η σύνδεση της JeVois με τον υπολογιστή πραγματοποιείται μέσω καλωδίου Micro USB to USB. Με αυτόν τον τρόπο ο χρήστης μπορεί να λαμβάνει τα δεδομένα εξόδου της κάμερας για να προχωρήσει σε επεξεργασία τους κλπ.



Εικόνα 7: Το περιβάλλον διαπροσωπείας του JeVois Inventor

Επικοινωνία με τον βραχίονα:

Η κάμερα ώντας συνδεδεμένη με θύρα USB, μπορεί να επικοινωνήσει με τον υπολογιστή. Κατά τη λειτουργία ενός module της JeVois, υπάρχει η επιλογή να αποστέλλονται 4-pin ή USB μηνύματα (Module output και log output) μέσω της θύρας USB. Για την εργασία αυτή δημιούργησα ένα πρόγραμμα στην Python όπου εκτελώ τις εξής λειτουργίες:

- Αρχικά με τη μέθοδο connect_to_camera δημιουργώ τη σύνδεση μεταξύ του υπολογιστή και της JeVois κάμερας. Βρίσκω τη συνδεδεμένη κάμερα, εαν υπάρχει, και ορίζω τις παραμέτρους της, frame width, frame height, frames per second και mode ανάλογα με το module που θα χρησιμοποιήσω (για τη λειτουργία αυτή υπεύθυνο είναι το Object Tracker). Για τις παραμέτρους κάμερας κάθε module της JeVois συμβουλεύτηκα την επίσημη σελίδα της.
- Έπειτα από τη σύνδεση με την κάμερα, επιχειρώ να δημιουργήσω τη σύνδεση με την σειριακή θύρα. Στη συνάρτηση connect_to_serial καθορίζω τη θύρα της JeVois στον υπολογιστή και ορίζω τις μεταβλητές της σύνδεσης ανάλογα. Θέτω ως mapping το Module που θα χρησιμοποιήσω, ως serout και serlog τη θύρα USB, θέτω το serstyle στο Normal και κάνω flush το serial buffer. Η επιλογή και ανάθεση του module στην κάμερα μπορεί να γίνει και από τις μεταβλητές mapping. Η εντολή listmappings επιστρέφει τη λίστα όλων των διαθέσιμων αντισοιχίσεων βίντεο, τον αριθμό δηλαδή που αντιστοιχεί σε κάθε module, και οι εντολές setmapping και setmapping2

εκτελούν τη χαρτογράφηση όταν δεν υπάρχει ροή βίντεο. Στην εργασία αυτή χρησιμοποίησα τυπικά την εντολή setmapping2 αλλά η αντιστοίχηση σε πραγματικό χρόνο όταν υπάρχει ροή βίντεο συμβαίνει στην μέθοδο connect_to_camera όπου ορίζονται οι παραμέτροι στης κάμερας, όχι στη σειριακή σύνδεση. Στον πίνακα που ακολουθεί φαίνονται οι μεταβλητές που χρησιμοποιήθηκαν με τη σημασία τους [10].

Μεταβλητή	**Περιγραφή**
srange	Εύρος τιμών κορεσμού για το παράθυρο HSV.
serstyle	Μορφή μηνυμάτων εξόδου. (Terse / Normal / Detail / Fine).
serout	Προώθηση μηνυμάτων εξόδου στη σειριακή θύρα που καθορίζει η μεταβλητή.
serlog	Προώθηση μηνυμάτων καταγραφής στη σειριακή θύρα που καθορίζει η μεταβλητή.
listmappings	Λίστα όλων των διαθέσιμων αντιστοιχίσεων βίντεο.
setmapping	Ορίζει τη χαρτογράφηση βίντεο. Είναι δυνατή μόνο όταν δεν υπάρχει ροή βίντεο. Παράμετρος: <num> : αριθμός χαρτογράφησης.
setmapping2	Ορίζει τη χαρτογράφηση βίντεο χωρίς USB-out. Είναι δυνατή όταν δεν υπάρχει ροή βίντεο. Πραμέτροι: <CAMmode> <CAMwidth> <CAMheight> <CAMfps> <Vendor> <Module>.

Πίνακας 1: Μεταβλητές σειριακών μηνυμάτων εξόδου

Πέραν από τις μεταβλητές που άλλαξα μέσω του κώδικα αυτού, με τη βοήθεια του λογισμικού της κάμερας, JeVois Inventor, επεξεργάστηκα και το script configuration αρχείο του Module Object Tracker ως εξής:

Αρχείο script.cfg:

```
# Configuration script for ObjectTracker module.
# Set camera to fixed color balance, gain, and exposure, so that we get more reliable colors than we would obtain under automatic mode:
setcam autowb 1
setcam autogain 0
setcam autoexp 0
setcam redbal 110
```

setcam bluebal 170
setcam gain 16
setcam absexp 500

Detect a light blue flash drive:
setpar hrange 95...120
setpar srange 100...255
setpar vrange 114...253

Send info log messages to None, send serial strings from module to Hard serial port:
setpar serlog None
setpar serout USB
setpar serstyle Normal

Apply high gain to our pan/tilt servos, sending the commands below to our Arduino over the Hard serial port that we configured above to handle the serout messages. The Arduino controlling the pan/tilt servos will receive and parse these commands, and will set the servo gains:
serout PANGAIN 400
serout TILTGAIN 300

Αφού καθιερώσω τη σύνδεση με την κάμερα και τη σειριακή θύρα, παίρνω frame ανά frame την εικόνα που παράγει η κάμερα για σκοπούς επιβεβαίωσης ότι λειτουργεί σωστά ο κώδικας, και διαβάζω τα σειριακά μηνύματα εξόδου που παράγει το module. Όταν εντοπίζει ένα αντικείμενο που πληρεί τα κριτήρια που καθορίζουν οι παραμέτροι του Module (είναι χρώματος μπλε), στέλνει μηνύματα του εξής τύπου [9] :

SERSTYLE	MESSAGE
TERSE	T2 x y
NORMAL	N2 id x y w h
DETAIL	D2 id x1 y1 x2 y2 x3 y3 x4 y4
FINE	F2 id n x1 y1 … xn yn

Πίνακας 2 Τύποι μηνυμάτων module output

Ως (x,y) ορίζονται οι συντεταγμένες (τυποποιημένη 2D θέση) του κέντρου του αντικειμένου που εντοπίστηκε, w και h το πλάτος και ύψος αντίστοιχα του τετραγώνου περιγράμματος που σχηματίζεται γύρω από το αντικείμενο που εντοπίστηκε, id η συμβολοσειρά κειμένου που περιγράφει τι είναι το αναφερόμενο αντικείμενο, x1,y1 ... x4,y4 οι τυποποιημένες συντεταγμένες x,y των 4 γωνιών ενός ορθογωνίου οριοθέτησης γύρω από το αναφερόμενο αντικείμενο., και τέλος, x1,y1 ... xn,yn οι τυποποιημένες συντεταγμένες x,y των n κορυφών ενός οριοθετημένου πολυγώνου γύρω από το αναφερόμενο αντικείμενο. Η τιμή n μπορεί να διαφέρει από αντικείμενο σε αντικείμενο.
Στην περίπτωση του Module Object Tracker, η τιμή του id για κάθε αντικείμενο που εντοπίζεται είναι «blob».

Στην εργασία αυτή επέλεξα να χρησιμοποιώ μηνύματα της μορφής Normal.
Ενσωμάτωσα στο module που εκτελώ (Object Tracker), μέσω του κώδικα μου, ένα window διαστάσεων 120x120 και εκτελώ την αναγνώριση ενός αντικειμένου κατά την ανίχνευση του εντός του πλαισίου του window.
Λόγω του ότι δεν χρησιμοποίησα κάποιο άλλο sensor για αναγνώριση της θέσης των αντικειμένων στον ιμάντα, με την επιπρόσθετη αυτή λειτουργία στο module δίνω στην κάμερα τον ρόλο του αισθητήρα και της αναθέτω την ανίχνευση των αντικειμένων. Σε περισσότερη λεπτομέρεια, με τη δημιουργία του window στην εικόνα της κάμερας, ορίζω το πλαίσιο εστίασης (περιβάλλον ενδιαφέροντος). Με την προσθήκη ελέγχων, έθεσα στην κάμερα να στέλνει θετικά σειριακά μηνύματα εξόδου μόνο όταν το ανιχνεύσιμο αντικείμενο βρίσκεται εντός του πλαισίου (window). Αλλιώς στέλνει το μήνυμα «timeout». Εαν υπάρχει κάποιο πρόβλημα, και το σειριακό μήνυμα εξόδου είναι άλλο, η κύρια συνάρτηση επιστρέφει με μήνυμα λάθους. Μέσω του κώδικα σχηματίζεται ένα επιπλέον μπλε κουτί (περίγραμμα) γύρω από ένα αντικείμενο (κύβο) όταν αυτό ανιχνευτεί, του οποίου ορίζω τις τέσσερις (4) συντεταγμένες γωνιών. Ο έλεγχος τοποθεσίας του ανιχνευμένου αντικειμένου γίνεται με βάση τις συντεταγμένες περιγράμματος του αντικειμένου, τις συντεταγμένες γωνιών του κουτιού και το frame της κάμερας. Έστησα την κάμερα σε σταθερό σημείο δίπλα στον βραχίονα με τέτοιο τρόπο ώστε η αριστερή άκρη του πλαισίου του window να συμπίπτει με την αριστερότερη άκρη του περιγράμματος του τετραγώνου γύρω από το ανιχνεύσιμο αντικείμενο, όταν αυτό βρίσκεται στην θέση κάτω από τον βραχίονα.

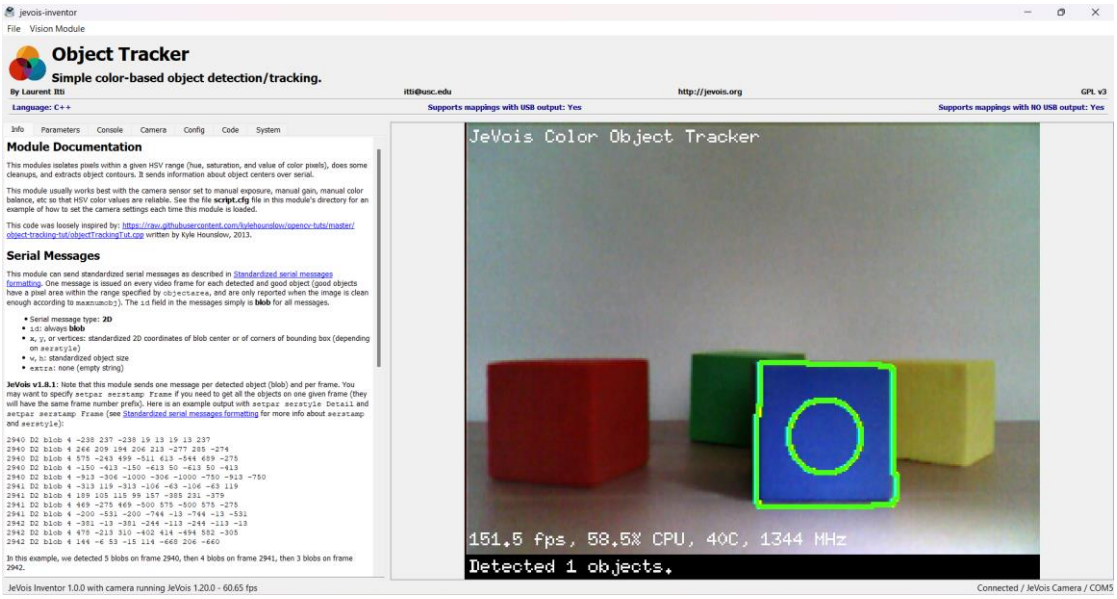
Οι συντεταγμένες γωνιών του περιγράμματος του αντικειμένου που ανιχνεύει η κάμερα παράγονται από την επεξεργασία των συντεταγμένων του κέντρου του αντικειμένου, το οποίο επιστρέφει η κάμερα στο μήνυμα εξόδου της, με την τιμή του ύψους και πλάτους του που επίσης επιστρέφει η κάμερα. Έπειτα από την εύρεση των 4 αυτών σημείων, υπάγονται σε κανονικοποίηση με βάση το frame που χρησιμοποιεί το module ώστε να ανταποκρίνονται σε αυτό. Αυτό συμβαίνει γιατί τα σειριακά μηνύματα εξόδου είναι προσαρμοσμένα στο default frame που ορίζει η JeVois το οποίο είναι [-1000, 1000] για τον άξονα των x και [-750, 750] για τον άξονα των y.
Χρησιμοποίησα τη φόρμουλα της γραμμικής παρεμβολής:

$$\textbf{\textit{scaled\_value}} = ((original\_value - original\_min) / (original\_max - original\_min)) * (new\_max - new\_min) + new\_min.$$

Αφού αλλάξω τις συντεταγμένες μέσω κανονικοποίησης, συγκρίνω τις συντεταγμένες του εντοπισμένου αντικειμένου με τις συντεταγμένες του παραθύρου μου για να ελέγξω εαν το αντικείμενο βρίσκεται εντός του παραθύρου. Εαν ισχύει, εμφανίζω το αντίστοιχο μήνυμα, αλλιώς, εξακολουθώ να αγνοώ το αντικείμενο.

2.6 Object Tracker module

Το module Object Tracker [13] (Ανιχνευτής Αντικειμένων) προσφέρει την ανίχνευση και παρακολούθηση αντικειμένων με βάση το χρώμα τους. Δημιουργήθηκε από τον Laurent Itti και είναι γραμμένο σε C++.
Το module αυτό απομονώνει εικονοστοιχεία (pixels) που βρίσκονται εντός ενός δεδομένου εύρους HVS (απόχρωση, κορεσμός, τιμή εικονοστοιχείων χρώματος), εκτελεί εκκαθαρίσεις και εξάγει τα περιγράμματα των αντικειμένων που ανιχνεύει. Έξοδος του είναι η συντεταγμένη του κέντρου (x,y) ενός αντικειμένου και οι τιμές πλάτους (w) και ύψους του (h).



Εικόνα 8: Το module Object Tracking

2.7 Meross Energy Monitor

Χρησιμοποίησα τον Meross MS310 Smart Wifi Plug with Energy Monitor [3] καταμετρητή ενέργειας κατά την εκτέλεση των πειραμάτων που ακολουθούν. Ο συγκεκριμένος καταμετρητής είναι στην EU μορφή. Επικοινωνεί με το Meross app το οποίο μπορεί να εγκασταθεί σε οποιαδήποτε κινητή συσκευή. Μέσω της εφαρμογής γίνεται παρακολούθηση της ενεργειακής κατανάλωσης των συνδεδεμένων συσκευών και ανάλυση διαφόρων στατιστικών. Χρειάζεται απλά να συνδεθεί ο καταμετρητής με το κινητό και την εφαρμογή. Τα χαρακτηριστικά της συσκευής φαίνονται στον πίνακα που ακολουθεί.

INPUT	EU: 100-240V~ , 50/60Hz
OUTPUT	EU: 16A Max
ENERGY MONITOR	Yes
WIRELESS STANDARDS	IEEE 802.11 b/g/n, 2.4GHz, 1T1R
CERTIFICATIONS	EU/FR/UK: CE, RoHs, WEEE

Εικόνα 9: Meross EU Smart Plug with Energy Monitor

Κεφάλαιο 3

Σύστημα και Λειτουργία

3.1 Εφαρμογές εξοικείωσης με το Dobot Magician

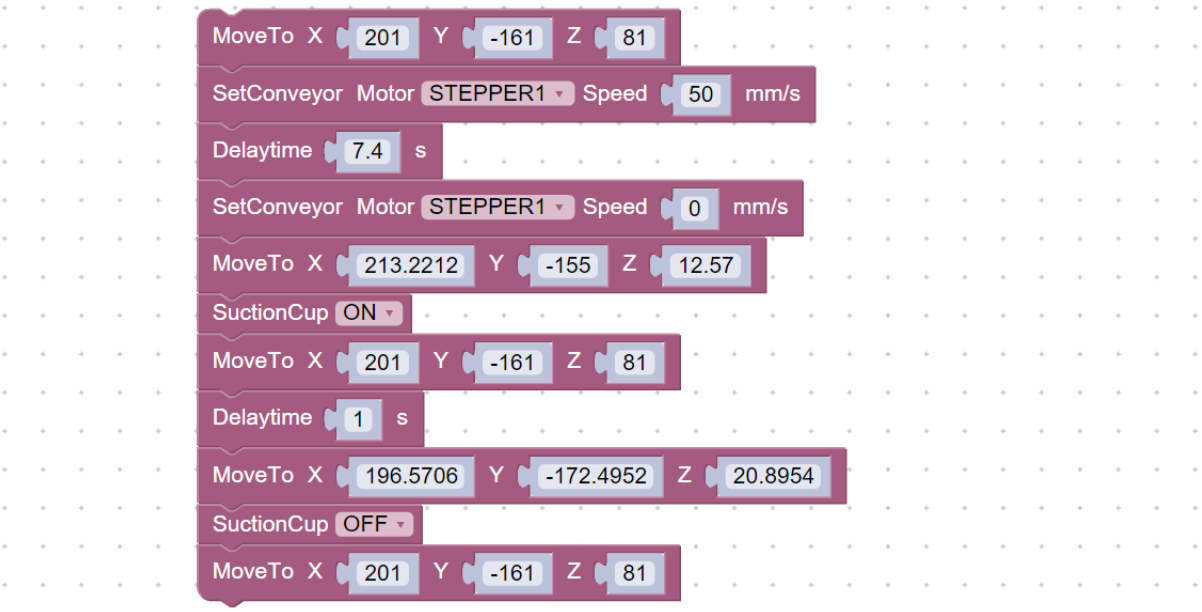
Για αρχική επαφή με τον ρομποτικό βραχίονα και ιμάντα, ανέπτυξα κάποιες απλές εφαρμογές για λειτουγία τους σε Blockly και Python.
Ακολουθούν κάποιες απ'αυτές τις εφαρμογές.

Εφαρμογή 1:
Απλή λειτουργία βραχίονα και ιμάντα. Ορίζω τις παραμέτρους θέσης, αρχικοποιώ τον βραχίονα και τοποθετώ ένα κύβο σε σταθερό σημείο στον ιμάντα. Ο βραχίονας βρίσκεται αρχικά στη θέση Home. Ο ιμάντας λειτουργεί και μεταφέρει τον κύβο σε θέση που όρισα (Pick) απ'όπου ο βραχίονας το παίρνει και μεταφέρει σε άλλη θέση (Release).
Κώδικας Python:

```
import math

xHome = 201
yHome = -161
zHome = 81
rHead = 108.4051

dType.SetPTPJointParams(api,200,200,200,200,200,200,200,200,0)
dType.SetPTPCoordinateParams(api,200,200,200,200,0)
```

```
dType.SetPTPJumpParams(api, 10, 200,0)
dType.SetPTPCommonParams(api, 100, 100,0)

#move arm to home position
dType.SetPTPCmd(api, 2, xHome, yHome, zHome, rHead, 1)

#Conveyor belt move
dType.SetEMotor(api, 0, 1, 10000,  0)
dType.dSleep(7400)
dType.SetEMotor(api, 0, 0, 0,  0)

#move arm
xPick=213.2212
yPick=-155
zPick=12.57

xRelease=196.5706
yRelease=-172.4952
zRelease=20.8954

dType.SetPTPCmd(api, 2, xPick, yPick, zPick, rHead, 1)
dType.dSleep(1500)
dType.SetEndEffectorSuctionCup(api, 1,  1, 0)
dType.SetPTPCmd(api, 2, xHome, yHome, zHome, rHead, 1)
dType.dSleep(1000)
dType.SetPTPCmd(api, 2, xRelease, yRelease, zRelease, rHead, 1)
dType.dSleep(1000)
dType.SetEndEffectorSuctionCup(api, 0,  1, 0)
dType.SetPTPCmd(api, 2, xHome, yHome, zHome, rHead, 1)
```

Ο αντίστοιχος κώδικας αναπτύχθηκε και σε Blockly:



Εφαρμογή 2:

Η εφαρμογή 2 αναπτύχθηκε για να εντάξει την εντολή dType.GetPose(api) του Dobot API η οποία επιστρέφει τις συντεταγμένες της θέσης του βραχίονα την στιγμή που γίνεται η κλήση της. Η διαφορά της με την εφαρμογή 1 είναι απλά ότι αντί να χρησιμοποιεί μεταβλητές για την κάθε θέση του βραχίονα, χρησιμοποιεί μόνο 1 μεταβλητή για κάθε άξονα.

Κώδικας σε Python:

```
import math

pos = dType.GetPose(api)
x = pos[0]
y = pos[1]
z = pos[2]
rHead = pos[3]

#setting the motion parameters
dType.SetPTPJointParams(api,200,200,200,200,200,200,200,200,0)
dType.SetPTPCoordinateParams(api,200,200,200,200,0)
dType.SetPTPJumpParams(api, 10, 200,0)
dType.SetPTPCommonParams(api, 100, 100,0)
```

```
#start position must be on the cube
dType.SetPTPCmd(api, 2, x, y, z+20, rHead, 1)
dType.dSleep(1000)
dType.SetPTPCmd(api, 2, x, y, z, rHead, 1)
dType.dSleep(1000)
dType.SetEndEffectorSuctionCup(api, 1,  1, 0)
dType.SetPTPCmd(api, 2, x, y, z+20, rHead, 1)
dType.dSleep(1000)
dType.SetPTPCmd(api, 2, x, y, z, rHead, 1)
dType.dSleep(1000)
dType.SetEndEffectorSuctionCup(api, 0,  1, 0)
```

Εφαρμογή 3:

Η εφαρμογή 3 είναι μια επέκταση της εφαρμογής 1 με βρόγχο επανάληψης της λειτουργίας παραλαβής του κύβου που βρίσκεται στον ιμάντα, από τον βραχίονα και τοποθέτηση του σε άλλη θέση. Όπως και στην εφαρμογή 2, γίνεται χρήση και της θέσης του βραχίονα κατά την έναρξη της εφαρμογής.

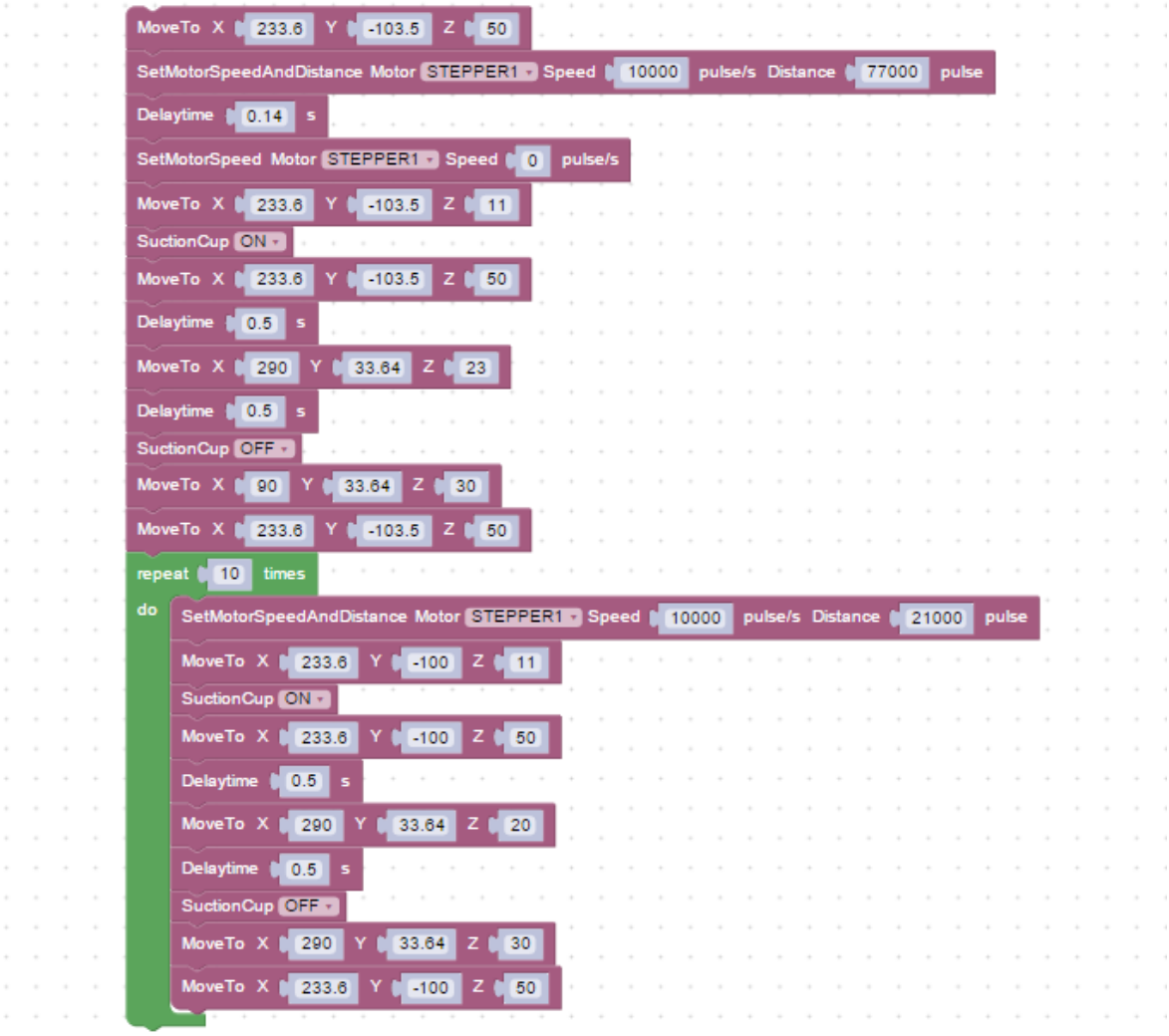
Κώδικας Python:

```
#T1
import math

pos = dType.GetPose(api)
xHome = 233.6
yHome = -103.5
zHome = 50
rHead = pos[3]

dType.SetPTPJointParams(api,200,200,200,200,200,200,200,200,0)
dType.SetPTPCoordinateParams(api,200,200,200,200,0)
dType.SetPTPJumpParams(api, 10, 200,0)
dType.SetPTPCommonParams(api, 100, 100,0)
```

```
#move arm to home position
dType.SetPTPCmd(api, 2, xHome, yHome, zHome, rHead, 1)
dType.SetEndEffectorSuctionCup(api, 0,  0, 0)

#Conveyor belt move & stop at pick up point
dType.SetEMotorS(api, 0, 1, 10000, 75000, 0)
dType.dSleep(8000)
dType.SetEMotor(api, 0, 0, 0,  0)

#Cube pick up points
xPick=233.6
yPick=-103.5
zPick=11

xRelease=284
yRelease=33.64
zRelease=20

dType.SetPTPCmd(api, 2, xPick, yPick, zPick, rHead, 1)
dType.SetEndEffectorSuctionCup(api, 1,  1, 0)
dType.SetPTPCmd(api, 2, xHome, yHome, zHome, rHead, 1)
dType.SetPTPCmd(api, 2, xRelease, yRelease, zRelease, rHead, 1)
dType.dSleep(3300)
dType.SetEndEffectorSuctionCup(api, 0,  0, 0)
dType.SetPTPCmd(api, 2, xHome, yHome, zHome, rHead, 1)
dType.dSleep(1000)
#Loop
yPick=-100
yHome = -100

for i in range(10):
    dType.SetEMotorS(api, 0, 1, 10000, 19100, 0)
```

```
dType.SetEMotor(api, 0, 0, 0,  0)
dType.SetPTPCmd(api, 2, xPick, yPick, zPick, rHead, 1)
dType.dSleep(3000)
dType.SetPTPCmd(api, 2, xPick, yPick, zPick, rHead, 1)
dType.SetEndEffectorSuctionCup(api, 1,  1, 0)
dType.SetPTPCmd(api, 2, xHome, yHome, zHome, rHead, 1)
dType.SetPTPCmd(api, 2, xRelease, yRelease, zRelease, rHead, 1)
dType.dSleep(3300)
dType.SetEndEffectorSuctionCup(api, 0,  1, 0)
dType.SetPTPCmd(api, 2, xHome, yHome, zHome, rHead, 1)
```

Αντίστοιχος κώδικας σε Blockly:



3.2 Διάταξη συστήματος

Εικόνα 10 Πειραματική διάταξη

Για την εκτέλεση των πειραμάτων χρησιμοποιήθηκε η πιο πάνω διάταξη. Τα αντικείμενα (κύβοι) τοποθετούνται με σειρά στην δεξιά άκρη του ιμάντα και ο ιμάντας τα μετακινεί προς την αριστερή του μεριά και τον βραχίονα. Η κάμερα είναι τοποθετημένη σε σταθερή θέση ώστε να εντοπίζει τους κύβους όταν φτάνουν στο επιθημητό σημείο. Έπειτα από τον εντοπισμό ενός αντικειμένου, ο ιμάντας σταματά να προχωράει και ο βραχίονας παίρνει το αντικείμενο για να το μεταφέρει σε θέση πιο πίσω απ΄αυτή που είναι το αντικείμενο. Δηλαδή σε θέση πιο αριστερά πανω στον ιμάντα.

3.3 Λειτουργία

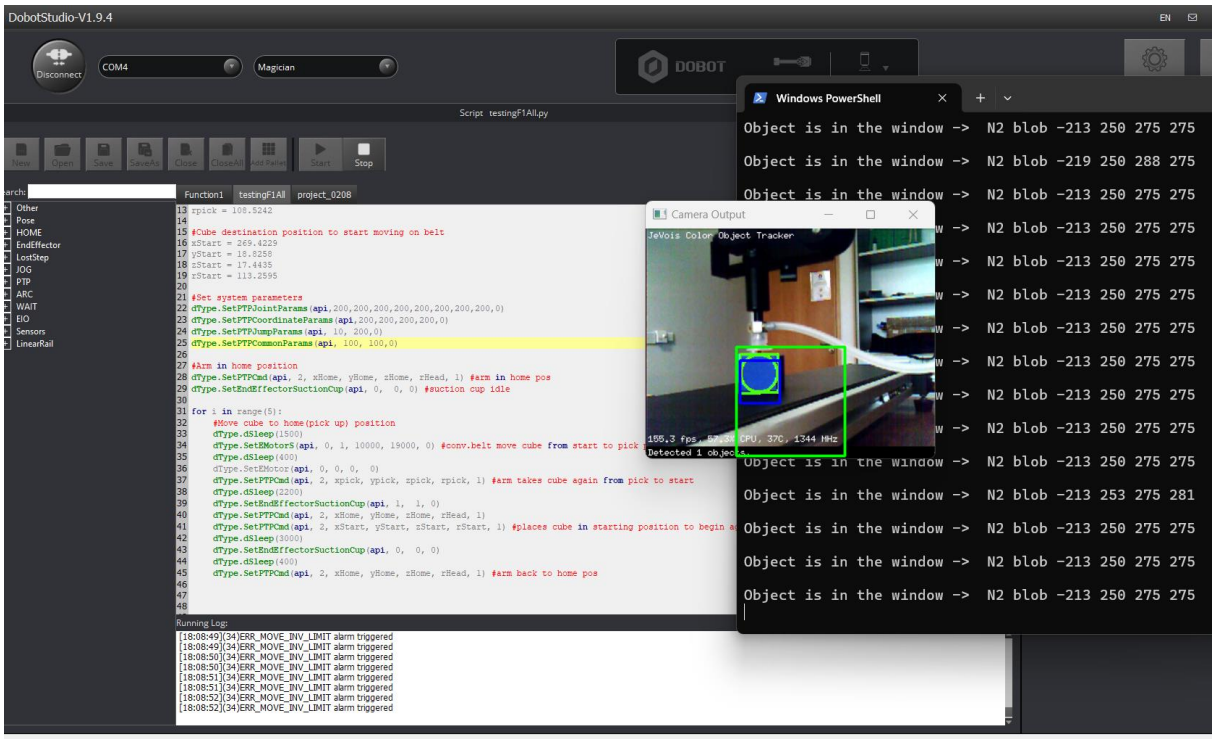
Η λειτουργία που έχω αναπτύξει, αποτελεί συνεργασία του πειραματικού εξοπλισμού για μεταφορά και αναγνώριση αντικειμένων.

Η κάμερα καταγράφει καθόλη τη διάρκεια στο περιβάλλον ενδιαφέροντος, ψάχνοντας να αναγνωρίσει αντικείμενα. Στον ιμάντα που είναι αρχικά ακίνητος, τοποθετούνται κύβοι σε σειρά, με απόσταση μεταξύ τους, μεγαλήτερη από το μήκος του frame της κάμερας. Ο βραχίονας βρίσκεται στην «Home», αρχική του θέση.

Με την έναρξη της λειτουργίας, ο ιμάντας ξεκινά να κινείται προς τα αριστερά με σταθερή ταχύτητα, μεταφέροντας τους κύβους. Όταν ένας κύβος εισέλθει στο πεδίο της κάμερας, τον καταγράφει. Η κάμερα εκτελεί τους απαραίτητους ελέγχους και την στιγμή που εντοπίζει ένα μπλε κύβο εντός του window που θέσαμε, ο ιμάντας σταματά να κινείται και ο βραχίονας παίρνει τον κύβο και τον μεταφέρει σε σημείο στον ιμάντα, κοντά στην αρχική θέση του κύβου και εκτός της εμβέλειας της κάμερας. Η διαδικασία επαναλαμβάνεται μέχρις ότου ο χρήστης τερματήσει τη ροή του προγράμματος.

Ο κώδικας της λειτουργίας είναι γραμμένος στη γλώσσα Python και εκτελείται μέσω terminal. Για σκοπούς ελέγχου και επαλήθευσης σωστής λειτουργίας, στον υπολογιστή που συνδέεται με το σύστημα και τρέχει τον κώδικα, εμφανίζεται παράθυρο με την έξοδο της κάμερας JeVois και τη λειτουργία της, καθώς και μηνύματα εξόδου στην κονσόλα.

Ο σχετικός κώδικας βρίσκεται στο παράρτημα της εργασίας.



Εικόνα 11: Παράδειγμα λειτουργίας Dobot και κώδικα JeVois από διαφορετικά περιβάλλοντα

Κεφάλαιο 4

Πειράματα

4.1 Μεθοδολογία

Κατά την εκτέλεση των πειραμάτων μελέτησα την κατανάλωση ενέργειας του Dobot και της JeVois, και τα ποσοστά επιτυχίας των εργασιών τους με κάποιες παραμέτρους να αλλάζουν κάθε φορά.

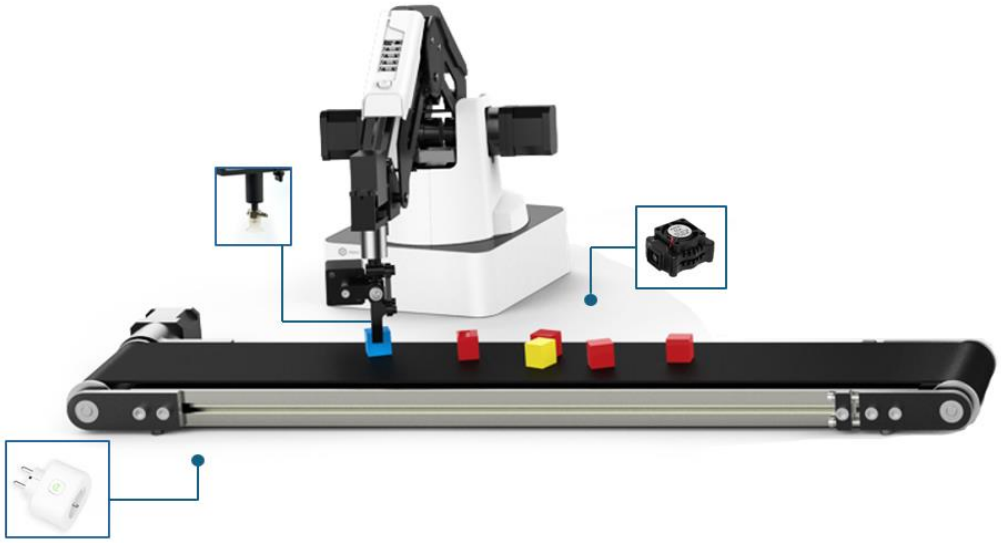
Τα πειράματα που εκτέλεσα αρχικά αφορούν την μεταβολή του χρόνου εκτέλεσης και του ποσοστού επιτυχίας ανίχνευσης της JeVois με μεταβλητό αριθμό επαναλήψεων της λειτουργίας.

Έπειτα διερεύνησα την κατανάλωση ενέργειας του Dobot και της JeVois ξεχωριστά κατά τις διάφορες καταστάσεις τους (ενεργοποιημένα χωρίς να εκτελούν λειτουργία, εκτελώντας λειτουργία).

Τέλος, παρατήρησα την ενεργειακή κατανάλωση και ποσοστά επιτυχίας κατά τη λειτουργία Dobot και JeVois μαζί (Λειτουργία εργασίας) με μεταβλητή ταχύτητα του Dobot ιμάντα.

4.2 Πειραματική διάταξη

Η πειραματική διάταξη των πειραμάτων αποτελείται από την διάταξη της λειτουργίας με επιπλέον τους αισθητήρες ενέργειας Meross για καταγραφή της ενεργειακής κατανάλωσης.



Εικόνα 12: Ολοκληρωμένη πειραματική διάταξη

4.3 Παραμέτροι μελέτης

Οι παραμέτροι που μελετήθηκαν κατά την εκτέλεση πειραμάτων ήταν η ταχύτητα του ιμάντα, το ποσοστό επιτυχίας αναγνώρισης αντικειμένων από την JeVois κάμερα και το ποσοστό επιτυχημένης ανασήκωσης και μεταφοράς κύβου από τον βραχίονα.

4.4 Αποτελέσματα

Πείραμα 1:

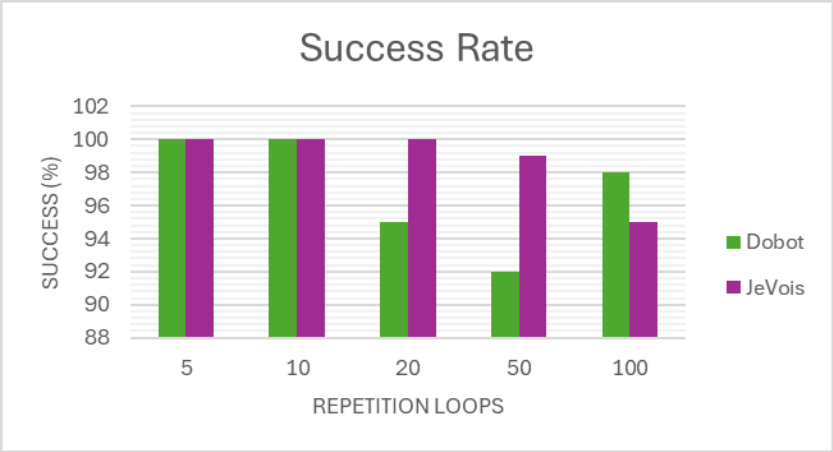
Μεταβολή χρόνου εκτέλεσης και ποσοστού επιτυχίας ανίχνευσης της JeVois με μεταβλητό αριθμό επαναλήψεων της λειτουργίας.

Στον πίνακα που ακολουθεί παρουσιάζονται τα αποτελέσματα (μετρήσεις) του πειράματος.

Στο πείραμα παρέμεινε σταθερή η ταχύτητα του ιμάντα στην τιμή 10000 (pulse/s).

ΑΡ. ΕΠΑΝΑΛΗΨΕΩΝ	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (S)	ΕΠΙΤΥΧΙΑ DOBOT (%)	ΕΠΙΤΥΧΙΑ JEVOIS (%)
5	40	100	100
10	77	100	100
20	151	95	100
50	380	92	99

Στην γραφική παράσταση που ακολουθεί φαίνεται το ποσοστό επιτυχίας του Dobot και της JeVois κάμερας αντίστοιχα για διαφορετικό αριθμό επαναλήψεων εκτέλεσης της Λειτουργίας.



Γραφική παράσταση 1: Ποσοστό επιτυχίας

Παρατηρήσεις:

Από την γραφική παράσταση του πειράματος παρατηρούμε ότι, καθώς ο αριθμός επαναλήψεων της εκτέλεσης αυξάνεται, το ποσοστό επιτυχίας και για το Dobot αλλά και για την JeVois, μειώνεται. Αυτό οφείλεται στο σφάλμα κατά την εκτέλεση λόγω της φύσης του πειράματος. Επίσης παρατηρούμε ότι για αριθμούς επαναλήψεων 20 και 50, η JeVois σημειώνει μεγαλύτερο ποσοστό επιτυχίας από το Dobot ενώ για 100 επαναλήψεις το Dobot σημειώνει μεγαλύτερη επιτυχία από την JeVois.

Κατά την εκτέλεση του πειράματος αυτού παρατηρήθηκε ότι η JeVois μπορεί να αποσυνδεθεί εαν λειτουργεί συνεχόμενα για μεγάλο χρονικό διάστημα (> 20 λεπτά). Συγκεκριμένα παρουσιάζεται να λειτουργεί όμως παύει να είναι συνδεδεμένη με τον υπολογιστή, ή ανιχνεύσιμη από αυτόν.

Πείραμα 2:

Κατανάλωση ενέργειας Dobot και JeVois ξεχωριστά κατά τις διάφορες καταστάσεις τους.

Οι τιμές ενέργειας ανά κατάσταση είναι κατά μέσο όρο. Η ενέργεια μετρήθηκε σε Watt (W), ο χρόνος εκτέλεσης σε δευτερόλεπτα (s) και η Επιτυχία σε ποσοστό (%). Ο αριθμός επαναλήψεων εκτέλεσης των πειραμάτων είναι ίσος με 100 και η ταχύτητα του ιμάντα κατά την εκτέλεση του πειράματος ήταν 10000 (pulse/s). Το σύνολο εκφράζει τη συνολική ποσότητα ενέργειας που καταναλώθηκε κατά την εκτέλεση του πειράματος, με βάση το mobile application παρακολούθησης Meross App.

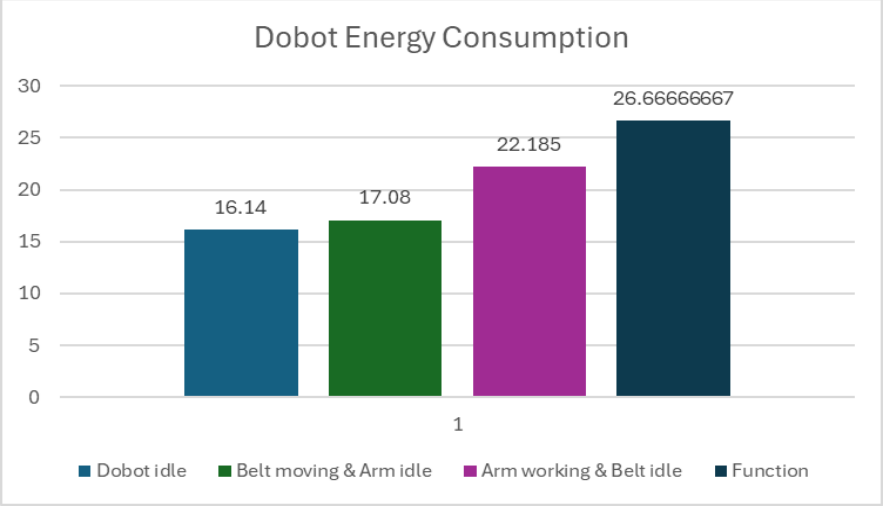
Dobot Magician:

ΚΑΤΑΝΑΛΩΣΗ ΕΝΕΡΓΕΙΑΣ	DOBOT ON	ΙΜΑΝΤΑΣ ΚΙΝΕΙΤΑΙ	ΒΡΑΧΙΟΝΑΣ ΔΟΥΛΕΥΕΙ	ΣΥΝΟΛΟ	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ	ΕΠΙΤΥΧΙΑ
ΕΝΕΡΓΕΙΑ	16.14 W	17.08 W	22.185 W	0.006 kWh	810 s	98 %

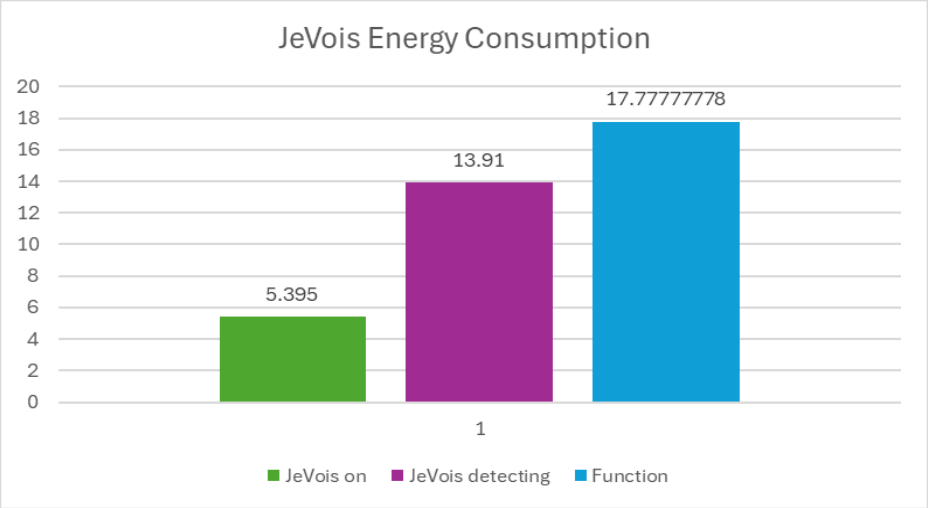
JeVois κάμερα:

ΚΑΤΑΝΑΛΩΣΗ ΕΝΕΡΓΕΙΑΣ	JEVOIS ON	JEVOIS ΑΝΙΧΝΕΥΕΙ	ΣΥΝΟΛΟ	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ	ΕΠΙΤΥΧΙΑ
ΕΝΕΡΓΕΙΑ	5.395 W	13.91 W	0.004 kWh	810 s	95 %

Στο πείραμα αυτό, λόγω του ότι η JeVois είναι πρώτα συνδεδεμένη στον υπολογιστή και ο υπολογιστής στον Meross αισθητήρα, οι τιμές που λήφθηκαν αφορούσαν την κατανάλωση ενέργειας από τον υπολογιστή και την JeVois μαζί. Έτσι αφαιρέθηκε το ποσό κατανάλωσης ενέργειας του υπολογιστής όταν δουλεύει χωρίς την κάμερα JeVois από το συνολικό ποσό για να λάβουμε τις τιμές που αφορούν μόνο την κάμερα. Ο υπολογιστής που χρησιμοποιήθηκε κατά το πείραμα είναι ο LENOVO Legion 5.
Από τις πιο πάνω μετρήσεις δημιουργήθηκαν οι ακόλουθες γραφικές παραστάσεις.



Γραφική παράσταση 2: Κατανάλωση ενέργειας από Dobot



Γραφική παράσταση 3: Κατανάλωση ενέργειας από JeVois

Παρατηρήσεις:

Οι καταστάσεις που μπορεί να βρεθεί το Dobot κατά την εκτέλεση της Λειτουργίας είναι οι εξής: Να είναι ενεργοποιημένο αλλά αδρανής (Dobot idle), να λειτουργεί ο ιμάντας και ο βραχίονας να είναι αδρανής (Belt working & Arm idle), να λειτουργεί ο βραχίονας και ο ιμάντας να είναι αδρανής (Arm working & Belt idle) και τέλος, να εκτελεί την Λειτουργία: Βραχίονας αδρανής καθώς ο ιμάντας κινείται, ιμάντας σταματά (αδρανής) και λειτουργεί ο βραχίονας και έπειτα η διαδικασία επαναλαμβάνεται (Function).

Οι καταστάσεις της JeVois κάμερας είναι οι εξής: Να είναι ενεργοποιημένη και να λειτουργεί χωρίς όμως να ανιχνεύει κάποιο αντικείμενο στο πλαίσιο της (JeVois on), να λειτουργεί και να ανιχνεύει αντικείμενο (JeVois detecting) και να εκτελεί την Λειτουργία: λειτουργεί χωρίς να ανιχνεύει, το αντικείμενο βρίσκεται στο window και η κάμερα ανιχνέυει και επαναλαμβάνεται η διαδικασία (Function).

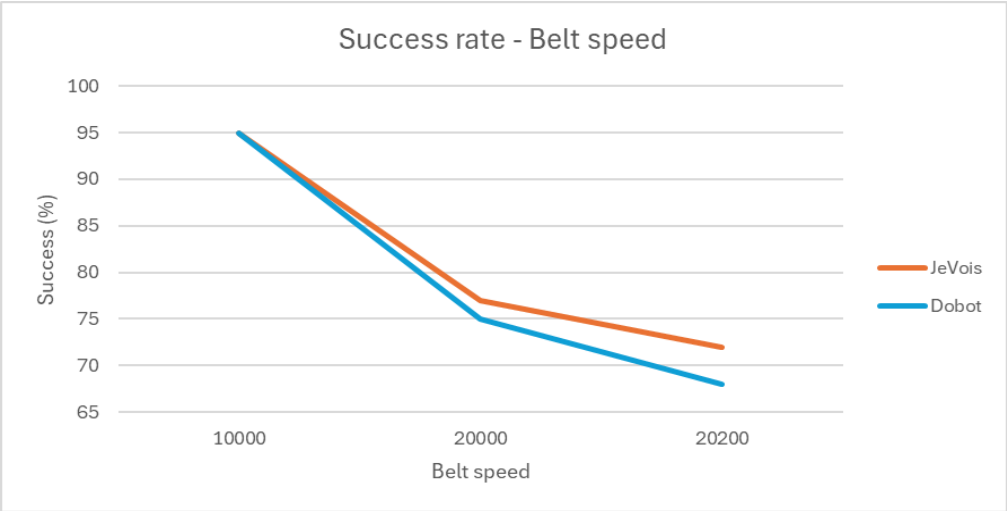
Από τις μετρήσεις κατανάλωσης ενέργειας για το Dobot και τη JeVois παρατηρούμε ότι το Dobot καταναλώνει περισσότερη ενέργεια για να εκτελέσει της Λειτουργίας απότι η JeVois. Επίσης, ο βραχίονας του Dobot καταναλώνει περισσότερη ενέργεια από τον ιμάντα, και η JeVois αυξάνει την κατανάλωση της κατά την ανίχνευση αντικειμένων.

Πείραμα 3:

Ποσοστά επιτυχίας κατά τη λειτουργία Dobot και JeVois μαζί (Λειτουργία εργασίας) με μεταβλητή την ταχύτητα του Dobot ιμάντα.

Ταχύτητα ιμάντα	*Αρ. επαναλήψεων*	*Επιτυχία JeVois (%)*	*Επιτυχία Dobot (%)*
10000	100	95	95
20000	100	77	75
20200	100	72	68

Οι τιμές του πίνακα εκφράζονται γραφικά στην γραφική παράσταση που ακολουθεί.



Γραφική παράσταση 4: Ποσοστό επιτυχίας με μεταβλητή ταχύτητα ιμάντα

Παρατηρήσεις:

Από την γραφική παράσταση παρατηρούμε ότι καθώς αυξάνεται η ταχύτητα του ιμάντα το ποσοστό επιτυχίας του Dobot καθώς και της JeVois μειώνεται. Αυτό οφείλεται στο ότι ο ιμάντας μεταφέρει τους κύβους γρηγορότερα, φτάνοντας σε ένα κατώφλι όπου η ταχύτητα μεταφοράς των κύβων υπερβαίνει τον χρόνο αντίδρασης της κάμερας και αυτή δεν προλαμβαίνει να ανιχνεύσει τους κύβους στη σωστή θέση για να παραλαβούν σωστά από τον βραχίονα. Ο χρόνος αντίδρασης αυτός της κάμερας αφορά τον χρόνο εκτέλεσης των εντολών του κώδικα που εκτελούν την ανίχνευση.

Παρατηρούμε επίσης ότι η JeVois έχει μεγαλύτερο ποσοστό επιτυχίας από τον βραχίονα για εύρος τιμών ταχύτητας ιμάντα [10000, 20200].

Η μέγιστη ταχύτητα ιμάντα ήταν η τιμή 20200. Έπειτα από αυτήν, ο ιμάντας δεν μετακινήτουν κατά την εκτέλεση της Λειτουργίας.

Η συνολική ταχύτητα του συστήματος δεν μειώνεται μόνο από την αύξηση της ταχύτητας του ιμάντα, χρειάζεται να αυξηθεί ανάλογα και η ταχύτητα του βραχίονα.

4.4.1 Προβλήματα

Κατά την εκτέλεση των αλγορίθμων και των πειραμάτων στο σύστημα, παρατηρήθηκαν τα εξής προβλήματα:

I. Το Dobot Magician «κόλλησε» σε αναπάντεχο λάθος (error) κατά τη λειτουργία του.

Κατά τη λειτουργία του βραχίονα, εμφάνισε error μήνυμα και σταμάτησε να λειτουργεί. Το error μήνυμα είναι το ακόλουθο: (34) ERR_MOVE_INV_LIMIT alarm triggered. Για να επιστρέψει στην κανονική του λειτουργία ο βραχίονας, πρέπει να επανεκκινήσει από το Reset κουμπί στο πλάι του ή να απενεργοποιηθεί και να ενεργοποιηθεί ξανά.

II. Η JeVois κάμερα έπαυε να λειτουργεί εντελώς μετά την επεξεργασία των περιεχομένων της κάρτας μνήμης της.

Κατά την έρευνα για χρήση του module Object Detection της JeVois υπήρξε η ανάγκη για εισαγωγή δεδομένων (φωτογραφιών) στην κάρτα μνήμης της JeVois. Η επίσημη σελίδα της JeVois αναφέρει ότι ένα μέσο επίτευξης του σκοπού αυτού είναι η αφαίρεση της microSD κάρτας της κάμερας και σύνδεση της με τον υπολογιστή. Παρουσιάστηκε πρόβλημα έπειτα από την εισαγωγή δεδομένων στην κάρτα με αυτόν τον τρόπο, καθώς η κάρτα της κάμερας έπαψε να εντοπίζεται από τον υπολογιστή. Λύση αυτού του προβλήματος είναι αντί να αφαιρεθεί η κάρτα από την κάμερα για να επεξεργαστεί, μέσω του λογισμικού JeVois Inventor και του module, να γίνει enable η εξαγωγή της κάρτας μνήμης ως φάκελος στον υπολογιστή. Τότε μπορεί ο χρήστης να επεξεργαστεί την κάρτα μνήμης της κάμερας χωρίς τέτοιο πρόβλημα.

III. Το Dobot μπορεί να χάσει τον προσανατολισμό του.

Είναι σημαντικό να ενεργοποιείται και να τοποθετήτε με τον ίδιο τρόπο κάθε φορά το Dobot Magician για να διατηρεί τις ίδιες συντεταγμένες. Εαν δεν συμβεί αυτό, οι συντεταγμένες του βραχίονα θα αλλάξουν και εαν στο module υπάρχουν ορισμένες τιμές συντεταγμένων, θα είναι διαφορετικές από τις επιθυμητές. Υπάρχει η δυνατότητα εκτέλεσης της διαδικασίας «Homing» του Dobot για επαναπροσδιορισμό των συντεταγμένων.

IV. Σφάλμα στην κοινή λειτουργία Dobot και JeVois

Κατά τη διαδικασία υλοποίησης της Λειτουργίας, αναπτύχθηκαν ξεχωριστά κώδικες για διαχείρηση του Dobot και της JeVois. Και τα δύο προγράμματα ελέγθηκαν ως προς τη λειτουργία και το σφάλμα τους και βελτιστοποιήθηκαν όσο ήταν δυνατό. Παρατηρήθηκε όμως ότι ενώ ξεχωριστά τα δύο προγράμματα δούλευαν άψογα, στον κώδικα που τα συνδιάζει, η αποδοτικότητα της JeVois κάμερας μειώθηκε. Κατά τη διαδικασία αποσφαλμάτωσης διαπιστώθηκε ότι αυτή η παρατήρηση δεν οφείλεται σε σφάλμα του κώδικα, αλλά σε καθυστέρηση του σήματος προς την JeVois κατά την εκτέλεση των εντολών του Dobot API στον βραχίονα.

Κεφάλαιο 5

Συμπεράσματα

5.1 Συμπεράσματα πειραμάτων

Από τα πειράματα που εκτελέστηκαν, δημιουργήθηκαν τα ακόλουθα συμπεράσματα:

1. Η ενεργειακή κατανάλωση του Dobot Magician είναι μεγαλύτερη από την τιμή ενεργειακής κατανάλωσης της JeVois κάμερας. Πρακτικά όμως, αφού η κάμερα πρέπει να είναι συνδεδεμένη με τον υπολογιστή για να εκτελέσει την Λειτουργία, το σύνολο της κάμερας με τον υπολογιστή καταναλώνει περισσότερη ενέργεια από τον βραχίονα και ιμάντα για εκτέλεση της Λειτουργίας.
2. Με την αύξηση της ταχύτητας του ιμάντα, μειώνεται το ποσοστό επιτυχίας της λειτουργίας. Επομένως τα μεγέθη ταχύτητας ιμάντα και επιτυχίας είναι αντιστρόφως ανάλογα. Όταν αυξάνεται το ένα, μειώνεται το άλλο.
3. Σε προσπάθεια να μειωθεί ο συνολικός χρόνος εκτέλεσης της Λειτουργίας μπορεί να μειωθεί το ποσοστό επιτυχίας με αποτέλεσμα να μην εκτελεστεί σωστά, ή να χρειαστεί τον εναπομήναν χρόνο για να μετακινήσει τα αντικείμενα που έχασε.

5.2 Μελλοντική εργασία

Διαπίστωσα με ικανοποίηση ότι η παρούσα εργασία θα μπορούσε να εξελιχθεί περαιτέρω, με ποικίλους τρόπους. Θα μπορούσαν να δημιουργηθούν κι άλλες λειτουργίες στην παρούσα πειραματική διάταξη, καθώς και να προστεθούν εξαρτήματα που μπορούν να την

αναπτύξουν. Για παράδειγμα, θα μπορούσε να προστεθεί επιπλέον βραχίονας που να δουλεύει παράλληλα με τον υφιστάμενο και να μελετηθεί η μεταξύ τους συνεργασία. Ακόμη, θα μπορούσε να προστεθεί κι άλλη κάμερα που παράλληλα με την JeVois θα παρακολουθεί το περιβάλλον και να συμβάλλει στην αναγνώριση αντικειμένων. Θα μπορούσαν να αναπτυχθούν κι άλλα modules στη κάμερα JeVois που να εκτελούν κάποια λειτουργία σε συνεργασία με το Dobot.

Βιβλιογραφία

[1] Ιστοσελίδα Dobot, https://www.dobot-robots.com/

[2] Jim Hanson & Chris Hurd, "Programming in DobotStudio with Blockly", 2019.

[3] "Smart Wi-Fi Plug with Energy Monitor MSS310 EU", https://www.meross.com/en-gc/smart-plug/smart-plug-google-home/6

[4] Rockwell Anyoha, "The History of Artificial Intelligence" , *SITN Harvard Kenneth C. Griffin Graduate School of Arts and Sciences,* 28 (August 2017).

[5] "Introduction to AI Applications in Robotics", *University of San Diego*.

[6] Robotnik, "The rise of Machine Learning Robots: Explore machine learning in robotics", *Robotnik*, June 2023.

[7] Carl DeSalvo, "7 Reasons Why Edge Computing is the Future of Robotics", *LinkedIn,* February 2024.

[8] "What is edge computing?", IBM.

[9] "Standardized serial messages formatting", http://jevois.org/doc/UserSerialStyle.html

[10] "Command-line interface user guide", http://jevois.org/doc/UserCli.html

[11] OpenCV About page, https://opencv.org/about/

[12] "JeVois Inventor graphical user interface", http://jevois.org/doc/JeVoisInventor.html

[13] "Object Tracker", http://jevois.org/moddoc/ObjectTracker/modinfo.html

[14] Ιστοσελίδα JeVois, http://jevois.org/

[15] Mohsen Soori, Behrooz Arezoo, Roza Dastres, "Optimization of energy consumption in industrial robots, a review", *Cognitive Robotics,* Volume 3, Pages 142-157, 2023.

Παράρτημα : Κώδικας Λειτουργίας

```
import threading
import math
import DobotDllType as dType
import cv2
import serial


def connect_to_serial():
    try:
        ser = serial.Serial('COM3', 9600, timeout=1)
        # ser.write(b'setmapping 47\n')
        # ser.write(b'listmappings\n')
        ser.write(b'setmapping2 YUYV 320 240 60.0 JeVois ObjectTracker\n')
        ser.write(b'setpar serout USB\n')
        ser.write(b'setpar serlog USB\n')
        ser.write(b'setpar serstyle Normal\n')
        ser.flush()  # Flush the serial buffer
        return ser
    except serial.SerialException as e:
        print("Error: Unable to connect to the serial port.", e)
        return None


def connect_to_camera():
    try:
        # Search for connected cameras
        camera = cv2.VideoCapture(0)

        # Check if camera is opened
        if not camera.isOpened():
```

```
            print("Error: Unable to connect to the camera.")
            return None

        # Set camera parameters for ObjectTracker module
        camera.set(cv2.CAP_PROP_FRAME_WIDTH, 320)  # Set frame width
        camera.set(cv2.CAP_PROP_FRAME_HEIGHT, 254)  # Set frame height
        camera.set(cv2.CAP_PROP_FPS, 60)  # Set frame rate
        camera.set(cv2.CAP_PROP_FOURCC,  cv2.VideoWriter_fourcc(*'YUYV'))  #  Set
camera mode to YUYV

        return camera
    except Exception as e:
        print("Error: Unable to connect to the camera.", e)
        return None


# Main function to capture camera output and display USB serial signals
def main():

    # Connect to camera
    camera = connect_to_camera()

    if camera is None:
        return

    # Connect to serial port
    ser = connect_to_serial()

    if ser is None:
        return


    CON_STR = {
```

```
    dType.DobotConnect.DobotConnect_NoError:  "DobotConnect_NoError",
    dType.DobotConnect.DobotConnect_NotFound: "DobotConnect_NotFound",
    dType.DobotConnect.DobotConnect_Occupied: "DobotConnect_Occupied"
}


#Load Dll and get the CDLL object
api = dType.load()  #=> api = CDLL("./DobotDll.dll",  RTLD_GLOBAL)
#建立与 dobot 的连接
#Connect Dobot
state = dType.ConnectDobot(api, "COM4", 115200)[0] #=> state = [result, masterDevType, slaveDevType, fwName, fwVer, masterId, slaveId, connectInfo.masterDevInfo.runTime]
print("Connect status:",CON_STR[state])

if (state == dType.DobotConnect.DobotConnect_NoError):

    #Clean Command Queued
    dType.SetQueuedCmdClear(api)

    #Async Motion Params Setting
    dType.SetHOMEParams(api, 200, 200, 200, 200, isQueued = 1)
    dType.SetPTPJointParams(api, 200, 200, 200, 200, 200, 200, 200, 200, isQueued = 1)
    dType.SetPTPCommonParams(api, 100, 100, isQueued = 1)

    #Home/rest position
    xHome = 209.1146
    yHome = -96.2862
    zHome = 49.2509
    rHead = 108.5242

    #Cube pick up position (under home)
    xpick = 213.6985
```

```
ypick = -97.8225
zpick = 11.2346
rpick = 108.5242

#Cube destination position to start moving on belt
xStart = 270.1956 #269.4229
yStart = 22.5212 #18.8258
zStart = 17.4435
rStart = 111

#camera
while True:
    # Capture frame-by-frame
    ret, frame = camera.read()

    # Display the resulting frame. Convert the image to BGR color space before displaying
    bgr_frame = cv2.cvtColor(frame, cv2.COLOR_YUV2BGR_YUYV)

    # Define the size and position of the square window
    square_size = 120
    center_x = bgr_frame.shape[1] // 2
    center_y = (bgr_frame.shape[0] * 3) // 4
    half_size = square_size // 2
    top_left = (center_x - half_size, center_y + half_size)
    bottom_right = (center_x + half_size, center_y - half_size)

    # Draw the square window on the frame
    cv2.rectangle(bgr_frame, top_left, bottom_right, (0, 255, 0), 2)
    #

    # cv2.imshow('Camera Output', bgr_frame) #uncomment
```

```
        # Read USB serial data
        serial_data = ser.readline().decode()
        if serial_data:
            # print(serial_data) #prints the serial_data

            #dobot move belt
            dType.SetQueuedCmdClear(api)
            lastIndex = dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode,
xHome, yHome, zHome, rHead, isQueued = 1)[0]
            lastIndex = dType.SetEndEffectorSuctionCup(api, 0,  0, isQueued = 1)[0]
            dType.SetEMotorS(api, 0, 1, 10000, 19000, isQueued = 1)
            dType.SetQueuedCmdStartExec(api)

            #Edit serial module output data
            tok = serial_data.split() # Split the line into tokens:
            if len(tok) < 6:
                continue # Skip if timeout or malformed line:
            if tok[0] != 'N2' or tok[1] != 'blob':  #  if not a standardized "Normal 2D" message:
                print("timeout")

            else:
                try:
                    obj_x = float(tok[2])
                    obj_y = float(tok[3])
                    obj_w = float(tok[4])
                    obj_h = float(tok[5])
                except ValueError:
                    continue

                #Calculate edge coordinates of found object square:
                objA = (int(obj_x - obj_w / 2), int(obj_y + obj_h / 2))
                objB = (int(obj_x + obj_w / 2), int(obj_y + obj_h / 2))
                objC = (int(obj_x - obj_w / 2), int(obj_y - obj_h / 2))
```

```
            objD = (int(obj_x + obj_w / 2), int(obj_y - obj_h / 2))

            #I adjust the coordinates to fit the new frame (old frame:[-1000, 1000], new frame: [320, 254])
            objA=(int(((objA[0]+1000)/(1000+1000))*(320-0)+0) , int(((objA[1]+750)/(750+750))*(254-0)+0))
            objB=(int(((objB[0]+1000)/(1000+1000))*(320-0)+0) , int(((objB[1]+750)/(750+750))*(254-0)+0))
            objC=(int(((objC[0]+1000)/(1000+1000))*(320-0)+0) , int(((objC[1]+750)/(750+750))*(254-0)+0))
            objD=(int(((objD[0]+1000)/(1000+1000))*(320-0)+0) , int(((objD[1]+750)/(750+750))*(254-0)+0))


            cv2.rectangle(bgr_frame, objA, objD, (255, 0, 0), 2)
            cv2.imshow('Camera Output', bgr_frame)


            #Calculate edge coordinates of window:
            wA = (center_x - half_size, center_y + half_size)
            wB = (center_x + half_size, center_y + half_size)
            wC = (center_x - half_size, center_y - half_size)
            wD = (center_x + half_size, center_y - half_size)

            #dobot
            # dType.SetQueuedCmdClear(api)
            # lastIndex = dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode, xHome, yHome, zHome, rHead, isQueued = 1)[0]
            # lastIndex = dType.SetEndEffectorSuctionCup(api, 0, 0, isQueued = 1)[0]
            # dType.SetEMotorS(api, 0, 1, 10000, 19000, isQueued = 1)
            # dType.SetQueuedCmdStartExec(api)
            # while lastIndex > dType.GetQueuedCmdCurrentIndex(api)[0]:
            #       dType.dSleep(100)
```

```
            # dType.SetQueuedCmdStopExec(api)

            #object is in the window area
            if(objA[0]>=wA[0] and objA[1]<=wA[1] and objB[0]<=wB[0] and objB[1]<=wB[1] and objC[0]>=wC[0] and objC[1]>=wC[1] and objD[0]<=wD[0] and objD[1]>=wD[1]):
                print("Object is in the window -> ", serial_data)

                #dobot
                dType.SetQueuedCmdStopExec(api)
                dType.SetQueuedCmdClear(api)

                dType.SetEMotor(api, 0, 0, 0,  isQueued = 1)
                dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode, xpick, ypick, zpick, rpick, isQueued = 1) #arm takes cube again from pick to start
                dType.dSleep(80)
                dType.SetEndEffectorSuctionCup(api, 1,  1, isQueued = 1)
                dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode, xHome, yHome, zHome, rHead, isQueued = 1)
                dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode, xStart, yStart, zStart, rStart, isQueued = 1) #places cube in starting position to begin again
                dType.dSleep(3000)
                dType.SetEndEffectorSuctionCup(api, 0,  0, isQueued = 1)
                dType.dSleep(400)
                lastIndex = dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode, xHome, yHome, zHome, rHead, isQueued = 1)[0] #arm back to home pos
                dType.SetEMotor(api, 0, 1, 10000, isQueued = 1) #19000


                #Start to Execute Command Queue
                dType.SetQueuedCmdStartExec(api)

                #Wait for Executing Last Command
```

```
                while lastIndex > dType.GetQueuedCmdCurrentIndex(api)[0]:
                    dType.dSleep(100)

                #Stop to Execute Command Queued
                dType.SetQueuedCmdStopExec(api)
                #loop


            # Check for exit command
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break

        # Release the camera, close serial port, and close all OpenCV windows
        camera.release()
        ser.close()
        cv2.destroyAllWindows()


    #Disconnect Dobot
    dType.DisconnectDobot(api)
    print("Disconnected Dobot\n")


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