ΠΑΝΕΠΙΣΤΗΜΙΟ ΣΤΕΡΕΑΣ ΕΛΛΑΔΑΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΙΑΤΡΙΚΗ

Κατασκευή και προγραμματισμός ρομπότ

Χαλκίδου Ναταλία

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ
Υπεύθυνος
Πλαγιανάκος Βασίλειος
Επίκουρος Καθηγητής

Λαμία, 2013

ΠΑΝΕΠΙΣΤΗΜΙΟ ΣΤΕΡΕΑΣ ΕΛΛΑΔΑΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΪΑΤΡΙΚΗ

Κατασκευή και προγραμματισμός ρομπότ

Χαλκίδου Ναταλία

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ
Υπεύθυνος
Πλαγιανάκος Βασίλειος
Επίκουρος Καθηγητής

Λαμία, 2013

Περίληψη

Στην παρούσα εργασία παρουσιάζεται η υλοποίηση ενός ρομπότ. Το ρομπότ που επιλέχθηκε να κατασκευαστεί είναι ένα σύστημα αυτόματου ελέγχου ισορροπίας και συνηθίζεται να αποκαλείται σύστημα σφαίρας σε πλάκα. Ο σκοπός του συστήματος είναι να ισορροπεί μια σφαίρα στο κέντρο μιας επίπεδης επιφάνειας.

Για την κατασκευή του συστήματος χρησιμοποιήθηκε το πακέτο ρομποτικών κατασκευών Lego Mindstorms και ένα smartphone κινητό με λειτουργικό Android. Το κυρίως σώμα της κατασκευής αποτελείται από τα "τουβλάκια", το IntelligentBrick και τους κινητήρες του πακέτου, ενώ το smartphone παρέχει στο σύστημα μηχανική όραση, για τον εντοπισμό της σφαίρας, και επεξεργαστική ισχύ για την εκτέλεση του προγράμματος.

Το σύστημα προγραμματίστηκε σε γλώσσα προγραμματισμού Java και το τελικό εκτελέσιμο αρχείο έχει τη μορφή μιας εφαρμογής Android. Το πρόγραμμα εκτελείται αποκλειστικά μόνο στο smartphone, το οποίο επικοινωνεί με το brick μέσω bluetooth, για να αποστέλλονται εντολές περιστροφής των κινητήρων. Το brick δεν εκτελεί κάποιο κώδικα αλλά λαμβάνει μόνο εντολές σε μορφή Direct commands που ερμηνεύονται από το Firmware του.

Abstract

This study presents the construction of a automatic balance control robot, usually called ball on a plate. Its purpose is to balance a ball in the center of a flat surface. For the construction of the system the Lego Mindstorms robotic construction kit was used. The intelligent software that controlled the robot was written and installed in an Android smartphone. The system consists of the Lego blocks, the Lego Intelligent Brick microcontroller, and the motors that move the surface. The smartphone provides machine vision to the system to detect the ball, as well as processing power to calculate the new orientation of the plate and the corresponding motor movements. The system was programmed in Java programming language and has the form of an Android application. The program is run exclusively on the smartphone, which communicates with the Intelligent Brick microcontroller via the Bluetooth protocol for sending commands to rotate the motors. The Intelligent Brick microcontroller does not run any code; it receives direct commands from the smartphone, which are interpreted by its Firmware.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή της εργασίας, κ. Βασίλειο Πλαγιανάκο κυρίως για την εμπιστοσύνη που μου έδειξε, και την υπομονή που έκανε κατά τη διάρκεια υλοποίησης της πτυχιακής εργασίας. Όπως επίσης και για την πολύτιμη βοήθεια και καθοδήγησή του, για την επίλυση διάφορων θεμάτων. Επίσης θα ήθελα να ευχαριστήσω τους καθηγητές της τριμελής επιτροπής, τον κ. Μάρκου Ευριπίδη και τον κ. Αναγνωστόπουλο Ιωάννη.

Ένα μεγάλο ευχαριστώ στον Ιωάννη Γιαννούλη για τη συνεχή στήριξη και συμπαράσταση. Τέλος, θέλω να ευχαριστήσω την οικογένειά μου για την ηθική και οικονομική συμπαράσταση όχι μόνο κατά τη διάρκεια της εκπόνησης της πτυχιακής μου εργασίας αλλά και καθ' όλη τη διάρκεια των σπουδών μου.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Τι είναι ένα ρομπότ

Το θέμα της πτυχιακής είναι η κατασκευή και ο προγραμματισμός ενός ρομπότ. Τι είναι όμως ένα ρομπότ; Στην ερώτηση αυτή δεν υπάρχει κάποια σαφής απάντηση. Ο όρος ρομπότ κάποιες φορές αναφέρεται σε μηχανική κατασκευή, που καθοδηγείται από ένα πρόγραμμα υπολογιστή ή ένα ηλεκτρονικό κύκλωμα, και άλλες φορές αναφέρεται σε ένα εικονικό πρόγραμμα, το οποίο αποκαλείται συνήθως bot ή crawler. Υπάρχουν πάρα πολλοί ορισμοί για τη λέξη ρομπότ, όπως οι παρακάτω:

- Σύμφωνα με τη διαδικτυακή εγκυκλοπαίδεια Wikipedia, "ρομπότ είναι μια μηχανική συσκευή που μπορεί να υποκαθιστά τον άνθρωπο σε διάφορες εργασίες και μπορεί να δράσει κάτω από τον απευθείας έλεγχο ενός ανθρώπου ή αυτόνομα κάτω από τον έλεγχο ενός προγραμματισμένου υπολογιστή"[1].

- Σύμφωνα με την εγκυκλοπαίδεια Britannica, "ρομπότ είναι οποιοδήποτε μηχανή αυτόματης λειτουργίας που αντικαθιστά την ανθρώπινη προσπάθεια, αν και ενδέχεται να μην μοιάζει στην εμφάνιση με τα ανθρώπινα όντα ή να μην εκτελεί τις λειτουργίες με ανθρωπόμορφο τρόπο" [2].

Η λέξη ρομπότ προέρχεται από την τσέχικη λέξη robota που σημαίνει καταναγκαστική εργασία και χρησιμοποιήθηκε πρώτη φορά από τον Τσέχο συγγραφέα Karel Capek το 1920 στο θεατρικό έργο R.U.R. (Rossum's Universal Robots) [3]. Η παράγωγη λέξη, ρομποτική, αναφέρεται στον κλάδο της τεχνολογίας που ασχολείται με το σχεδιασμό, την κατασκευή, τη λειτουργία και την εφαρμογή των ρομπότ, ενώ χρησιμοποιήθηκε σαν όρος για πρώτη φορά από το συγγραφέα Isaac Asimov [3].

1.2 Τα συστατικά ενός ρομπότ

Ένα ρομπότ αποτελείται από τρία μέρη: τους αισθητήρες (sensors), τα εξαρτήματα δράσης (effectors) και τη μονάδα ελέγχου (control unit).

Αισθητήρες:
Αισθητήρες είναι τα εξαρτήματα που παρέχουν στο ρομπότ τις πληροφορίες του περιβάλλοντος αλλά και του ίδιου του ρομπότ. Παράδειγμα, οι κάμερες και οι συσκευές sonar είναι αισθητήρες και προσφέρουν μηχανική όραση στο ρομπότ, έτσι το ρομπότ μπορεί να αντιλαμβάνεται τα εμπόδια στη διαδρομή του, να εντοπίζει κάποιο συγκεκριμένο αντικείμενο κτλ.. Ενώ οι αισθητήρες GPS παρέχουν στο ρομπότ τη δυνατότητα αντίληψης ως προς το που βρίσκεται το ίδιο, με βάση τις γεωγραφικές συντεταγμένες. Υπάρχει μεγάλη ποικιλία αισθητήρων όπως ήχου, φωτός, αφής, θερμοκρασίας, προσανατολισμού κ.ά..

Εξαρτήματα δράσης:
Τα εξαρτήματα δράσης εκτελούν ενέργειες πάνω στο περιβάλλον με σκοπό τη μεταβολή του ή τη μεταβολή του ίδιου του ρομπότ. Συνήθως είναι κινητήρες ή βραχίονες και δίνουν στο ρομπότ τη δυνατότητα μετακίνησης του ίδιου ή κάποιου άλλου αντικειμένου. Οι δυνατές κινήσεις του ρομπότ εκφράζονται σε βαθμούς ελευθερίας. Σε κάθε ανεξάρτητη κατεύθυνση προς την οποία μπορεί να κινηθεί ένα ρομπότ ή μια από τις συσκευές δράσης του, αντιστοιχεί ένας βαθμός ελευθερίας [4]. Σε γενικές γραμμές δηλώνει το πόσο ευκίνητο είναι ένα ρομπότ στο χώρο.

Μονάδα ελέγχου:
Μονάδα ελέγχου ενός ρομπότ ουσιαστικά είναι ένας υπολογιστής που επεξεργάζεται τα δεδομένα που λαμβάνει, μέσω των αισθητήρων, από το γύρω περιβάλλον και σχεδιάζει τις ενέργειες που θα εκτελέσει με τη βοήθεια των εξαρτημάτων δράσης, λαμβάνοντας υπόψη τους σκοπούς του ρομπότ που πρέπει να επιτελέσει.

1.3 Κατηγορίες ρομπότ

Τα ρομπότ μπορούν να διαχωριστούν σε τρεις μεγάλες κατηγορίες: τα χειριστήρια (manipulators), τα μετακινούμενα ή κινητά (mobile) και τα ανθρωποειδή (humanoid) ρομπότ. Ο διαχωρισμός αυτός γίνεται βάσει των μορφολογικών χαρακτηριστικών [5].

Χειριστήρια ρομπότ:

Τα χειριστήρια ρομπότ ή ρομποτικοί βραχίονες συνήθως είναι προσαρμοσμένοι σε μια σταθερή βάση. Η κίνηση του βραχίονα αποτελείται συνήθως από μια σειρά ελεγχόμενων αρθρώσεων που επιτρέπουν στο ρομπότ να χειρίζεται διάφορα αντικείμενα. Τα χειριστήρια ρομπότ αναφέρονται πολλές φορές και ως κατασκευαστικά ρομπότ (manufacturing robots) γιατί είναι ο συνηθέστερος τύπος βιομηχανικών ρομπότ κατασκευής-συναρμολόγησης αντικειμένων [5, 4].

Μετακινούμενα ρομπότ:

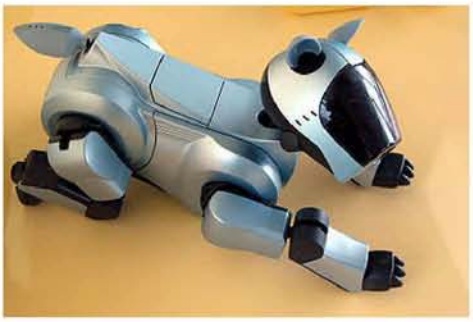
Με τον όρο μετακινούμενα ρομπότ χαρακτηρίζονται αυτά που μπορούν να μετακινούνται ελεύθερα σε ένα χώρο και να ενεργούν στα γύρω αντικείμενα. Χωρίζονται σε αυτόνομα μετακινούμενα ρομπότ και σε ρομπότ που δρουν κάτω από τον απευθείας έλεγχο του ανθρώπου [5].

Ανθρωποειδή ρομπότ:

Τα ανθρωποειδή ρομπότ είναι μια μίξη των δύο προηγούμενων κατηγοριών. Είναι αυτόνομα μετακινούμενα ρομπότ, εφοδιασμένα με βραχίονες και μοιάζουν σε εμφάνιση με τον άνθρωπο. Τα ρομπότ αυτά κατασκευάζονται κυρίως λόγω ερευνητικού και ψυχαγωγικού ενδιαφέροντος και ο σκοπός τους είναι να μιμούνται την ανθρώπινη συμπεριφορά [5, 4]. Το πρώτο ανθρωπόμορφο ρομπότ αναφέρεται στην αρχαία ελληνική μυθολογία, ονομαζόταν Τάλως και ήταν ένας μυθικός, γιγάντιος φύλακας της Κρήτης φτιαγμένος από χαλκό.

Το πιο δημοφιλές ρομπότ του είδους είναι το Asimo της εταιρίας Honda (Σχήμα 1.1). Επίσης ένα άλλο δημοφιλές ρομπότ, το AIBO της Sony έχει μορφή κατοικίδιου σκύλου αλλά φέρει παρόμοια χαρακτηριστικά με τα ανθρωποειδή ρομπότ (Σχήμα 1.2).

Σχήμα 1.1: *ASIMO, ένα ανθρωποειδές ρομπότ της Honda [6]*



Σχήμα 1.2: *AIBO, ρομπότ σε μορφή κατοικίδιου σκύλου της Sony [7]*

1.4 Πεδία εφαρμογής της ρομποτικής

Στις μέρες μας, η τεχνολογία της ρομποτικής έχει εισχωρήσει σχεδόν σε κάθε κλάδο απασχόλησης και έχει γίνει πλέον απαραίτητη στους περισσότερους από αυτούς. Μερικές εφαρμογές της ρομποτικής αναφέρονται παρακάτω [4]:

Σχήμα 1.3: *Το ερευνητικό διαστημικό ρομπότ της NASA [8]*

Σχήμα 1.4: *Da Vinci χειρουργικό ρομποτικό σύστημα [9]*

Βιομηχανία:

Τα ρομπότ χρησιμοποιούνται σε τομείς που απαιτούν δύσκολη ανθρώπινη εργασία και επιδέχονται ρομποτική αυτοματοποίηση. Τυπικές εφαρμογές των ρομπότ περιλαμβάνουν τη συναρμολόγηση, την τοποθέτηση τμημάτων, τη συγκόλληση, τη συσκευασία, την επιθεώρηση και τον έλεγχο, και όλα αυτά με μεγάλη ταχύτητα και ακρίβεια.

Επικίνδυνα περιβάλλοντα:

Τα ρομπότ έχουν βοηθήσει τους ανθρώπους στον καθαρισμό πυρηνικών αποβλήτων, όπως στο Τσερνομπίλ. Έχουν χρησιμοποιηθεί για τον αφοπλισμό βομβών. Γενικώς, κρίνονται απαραίτητα όπου επικρατούν αντίξοες συνθήκες για τον άνθρωπο.

Εξερεύνηση:

Τα ρομπότ έχουν πάει σε μέρη όπου είναι δύσκολη ή επικίνδυνη η προσπέλαση από ανθρώπους, όπως παράδειγμα η επιφάνεια του Άρη (Σχήμα 1.3) και ο βυθός του ωκεανού, με σκοπό τη συγκέντρωση δεδομένων και χαρτογράφηση για ερευνητικούς σκοπούς.

Ιατρική φροντίδα:

Ίσως η πιο σπουδαία εφαρμογή των ρομπότ είναι στην ιατρική. Τη σήμερον ημέρα, ένας χειρούργος μπορεί να πραγματοποιήσει χειρουργικές επεμβάσεις με τη βοήθεια ρομποτικών συστημάτων, κάνοντας πολύ

μικρές και ακριβείς τομές, μειώνοντας σημαντικά τον κίνδυνο για τους ασθενείς (Σχήμα 1.4). Επίσης στόχος της ιατρικής κοινότητας είναι η ανάπτυξη αποτελεσματικών εμφυτευμένων συσκευών και τεχνητών ρομποτικών άκρων, διευκολύνοντας την καθημερινή ζωή των ασθενών.

Προσωπική εξυπηρέτηση:
Τα ρομπότ εξυπηρέτησης βοηθούν τους ανθρώπους στην εκτέλεση των καθημερινών εργασιών. Χαρακτηριστικό παράδειγμα είναι η αυτόματη ηλεκτρική σκούπα.

1.5 Αντικείμενο της πτυχιακής εργασίας

Στην παρούσα εργασία υλοποιήθηκε ένα ρομπότ που έχει σκοπό να ισορροπεί μια σφαίρα στο κέντρο μιας επίπεδης επιφάνειας. Είναι μια σταθερή κατασκευή, η οποία κινεί μια πλάκα με τέτοιο τρόπο ώστε η σφαίρα, που βρίσκεται πάνω σε αυτή, να είναι πάντα στο κέντρο της. Η διαδικασία αυτή επιτυγχάνεται με την εντοπισμό της σφαίρας από μια κάμερα που βλέπει το πάνω μέρος της επιφάνειας, και την αποστολή κατάλληλων εντολών στους κινητήρες που περιστρέφουν, προς δυο διαστάσεις, την επίπεδη επιφάνεια.

Το ρομπότ αυτό είναι ένα σύστημα αυτόματου έλεγχου ισορροπίας, δηλαδή είναι ένα ρομπότ, το οποίο λειτουργεί με ένα πρόγραμμα που τρέχει σε κάποια υπολογιστική μονάδα, χωρίς να δρα κάτω από τον έλεγχο κάποιου ανθρώπου / χειριστή. Πολλές φορές το ρομπότ αυτό αναφέρεται ως σύστημα αυτόματου ελέγχου σφαίρας σε πλάκα, και είναι μια παραλλαγή του συστήματος σφαίρας σε ράβδο, ενός ρομπότ δηλαδή που ισορροπεί τη σφαίρα πάνω σε μια ράβδο (ισορροπία σε μια διάσταση).

Το σύστημα σφαίρας σε πλάκα είναι αρκετά δύσκολο στη σχεδίαση και την υλοποίηση, γιατί είναι ένα σύστημα που επηρεάζεται από πολλές μεταβλητές που το καθιστούν ασταθές και αδύνατο στη πλήρη μαθηματική μοντελοποίηση. Η λύση στο ασταθές περιβάλλον είναι η χρήση της ανάδρασης (feedback). Ανάδραση ονομάζεται η ανατροφοδότηση της εξόδου ενός συστήματος στην είσοδο, όπου η έξοδος συνήθως υπολογίζεται με την βοήθεια των αισθητήρων. Στην περίπτωσή μας ο αισθητήρας είναι η κάμερα και η έξοδος του συστήματος είναι η θέση της σφαίρας πάνω στην επιφάνεια. Έτσι η πληροφορία για την θέση της σφαίρας ανατροφοδοτείται στο πρόγραμμα και υπολογίζεται η καινούρια κίνηση της επιφάνειας, για να επιτευχθεί ο σκοπός του συστήματος.

Για την κατασκευή του ρομπότ χρησιμοποιήθηκε το πακέτο ρομποτικών κατασκευών Lego Mindstorms NXT2.0. Είναι ένα πακέτο που περιλαμβάνει τα "τουβλάκια", με τα οποία συναρμολογείται το κυρίως σώμα ενός ρομπότ,

τους αισθητήρες, τους κινητήρες και το IntelligentBrick, το οποίο είναι μια μικρή υπολογιστική μονάδα ελέγχου και πηγή ενέργειας για τη ρομποτική κατασκευή. Επίσης για την υλοποίηση του ρομπότ χρησιμοποιήθηκε ένα κινητό smartphone με λειτουργικό android. Η συσκευή αυτή παρέχει επιπλέον αισθητήρες, όπως είναι η κάμερα που χρησιμοποιήθηκε, και περισσότερη υπολογιστική ισχύ.

Το ρομπότ προγραμματίστηκε σε γλώσσα προγραμματισμού Java και το εκτελέσιμο αρχείο έχει τη μορφή μιας εφαρμογής Android. Η κινητή συσκευή τρέχει τον κώδικα και αποστέλλει τις εντολές μέσω bluetooth στο brick, το οποίο στη συνέχεια κινεί τα εξαρτήματα δράσης. Οι εντολές που αποστέλλονται είναι σε μορφή Direct commands, κατάλληλα συνταγμένες για να ερμηνεύονται από το firmware του brick. Επίσης χρησιμοποιήθηκε η βιβλιοθήκη επεξεργασίας βίντεο πραγματικού χρόνου OpenCV, για τον εντοπισμό της σφαίρας.

1.6 Οργάνωση της πτυχιακής εργασίας

Το κεφάλαιο 2 εξηγεί τι είναι τα συστήματα αυτόματου ελέγχου και αναλύει το αυτόματο σύστημα σφαίρας σε πλάκα, από φυσική και μαθηματική άποψη. Επίσης περιγράφονται οι PID ελεγκτές που βοηθάνε στη αποτελεσματικότερη λειτουργία του συστήματος που υλοποιήθηκε.

Το κεφάλαιο 3 περιγράφει αναλυτικά το πακέτο Lego Mindstorms και το android smartphone που χρησιμοποιήθηκαν για την κατασκευή του συστήματος, καθώς επίσης παρουσιάζει τις βιβλιοθήκες και το λογισμικό που χρησιμοποιήθηκαν για τον προγραμματισμό του συστήματος.

Στο κεφάλαιο 4 σε πρώτη φάση παρουσιάζεται η διαδικασία της κατασκευής του ρομπότ και σε δεύτερη φάση αναλύονται τα βήματα του προγραμματισμού του ρομπότ.

Στο τελευταίο κεφάλαιο παρατίθενται τα συμπεράσματα, όπως και κάποιες πιθανές διορθώσεις και επεκτάσεις.

Στο παράρτημα Α΄ παρουσιάζονται αναλυτικά τα βήματα συναρμολόγησης της ρομποτικής κατασκευής. Στο παράρτημα Β΄ βρίσκεται ο τελικός πηγαίος κώδικας του ρομπότ. Στα παραρτήματα Γ΄ και Δ΄ παρατίθενται τα τεχνικά χαρακτηριστικά του IntelligentBrick και της κινητής συσκευής που χρησιμοποιήθηκαν και στο παράρτημα Ε΄ παρουσιάζεται το γραφικό περιβάλλον της εφαρμογής που ελέγχει την κατασκευή.

Κεφάλαιο 2

Περιγραφή και μαθηματική μοντελοποίηση του φυσικού συστήματος σφαίρας σε πλάκα

Στο κεφάλαιο αυτό θα εξηγηθούν τα συστήματα αυτόματου ελέγχου και έπειτα θα περιγραφούν τα συστήματα οφαίρας σε ράβδο (ball and beam system) και οφαίρας σε πλάκα (ball and plate system) με το τελευταίο να είναι και αντικείμενο της παρούσας πτυχιακής. Έπειτα, θα μοντελοποιηθεί μαθηματικά το σύστημα οφαίρας σε πλάκα και τέλος, θα εξηγηθούν οι ελεγκτές PID, που είναι οι σημαντικότεροι παράγοντες της επιτυχίας του συστήματος.

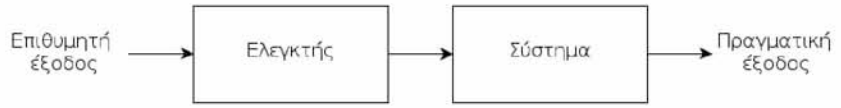
2.1 Σύστημα αυτόματου ελέγχου

Το σύστημα που θα υλοποιηθεί είναι ένα σύστημα αυτόματου ελέγχου, το οποίο είναι ένα σύνολο διασυνδεδεμένων στοιχείων και εξαρτημάτων που μπορούν να εξασφαλίσουν ικανοποιητική απόκριση, σε διάφορες συνθήκες λειτουργίας για διάφορα σήματα εισόδου (Σχήμα 2.1). Τα συστήματα ελέγχου χωρίζονται σε συστήματα ελέγχου ανοιχτού βρόγχου (open-loop systems) και συστήματα ελέγχου κλειστού βρόγχου (closed-loop systems) [10].

Σχήμα 2.1: *Απλή αναπαράσταση συστήματος ελέγχου*

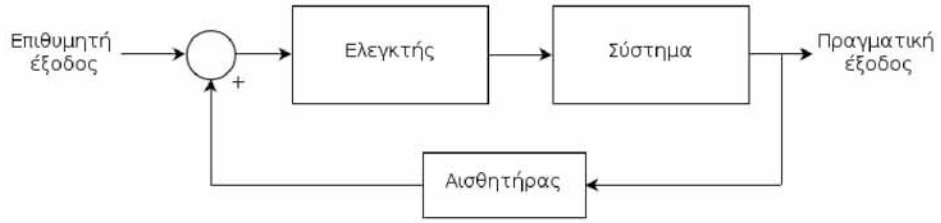
Στα συστήματα ανοιχτού βρόγχου χρησιμοποιείται ένας ελεγκτής (controller) για να ελέγξει απευθείας τις τιμές εξόδου χωρίς ανατροφοδότηση

(feedback). Τα συστήματα αυτά είναι εύκολα στην κατασκευή, δεν παρουσιάζουν προβλήματα αστάθειας, αλλά δεν έχουν πάντα ικανοποιητική απόκριση, ειδικά σε ακραίες συνθήκες λειτουργίας (Σχήμα 2.2) [10].



Σχήμα 2.2: *Αναπαράσταση συστήματος ελέγχου ανοιχτού βρόγχου*

Στα συστήματα κλειστού βρόγχου έχουμε επιπλέον, σε σχέση με τα συστήματα ανοιχτού βρόγχου, και τον κλάδο ανατροφοδότησης, ο οποίος συνήθως περνάει μέσα από έναν αισθητήρα (sensor). Η διαφορά μεταξύ της επιθυμητής εξόδου (εισόδου) και της πραγματικής εξόδου είναι το σφάλμα του συστήματος. Το σφάλμα αυτό ρυθμίζεται από τον ελεγκτή με στόχο την ελαχιστοποίησή του και βελτιστοποίηση της απόκρισης. Με τα συστήματα κλειστού βρόγχου επιτυγχάνεται μεγαλύτερη ακρίβεια ελέγχου του συστήματος (Σχήμα 2.3) [10].



Σχήμα 2.3: *Αναπαράσταση συστήματος ελέγχου κλειστού βρόγχου*

Ένα σύστημα αυτόματου ελέγχου πρέπει να ικανοποιεί ορισμένες απαιτήσεις, όπως:

Ευστάθεια:
Κατά πόσο η έξοδος του συστήματος δεν υπερβαίνει τα επιτρεπτά όρια.

Ακρίβεια:
Πόση είναι η απόκλιση μεταξύ της επιθυμητής και της πραγματικής τιμής εξόδου.

Ταχύτητα απόκρισης:
Πόσο γρήγορα ανταποκρίνεται το σύστημα.

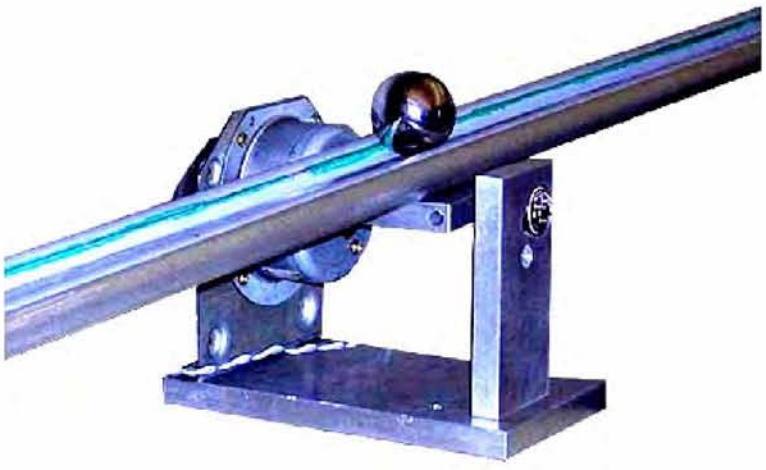
Ευαισθησία:
Πόσο εύκολα επηρεάζεται η έξοδος από τις μικρές μεταβολές της εισόδου.

Τέλος για την υλοποίηση ενός συστήματος αυτόματου ελέγχου υπάρχουν τα εξής βασικά στάδια:

- πλήρης κατανόηση του συστήματος,
- σχεδίαση και μαθηματική μοντελοποίηση του συστήματος,
- κατασκευή και προγραμματισμός του συστήματος,
- πειραματικοί έλεγχοι για την διαπίστωση της επιθυμητής λειτουργίας του συστήματος.

2.2 Περιγραφή του συστήματος σφαίρας σε ράβδο

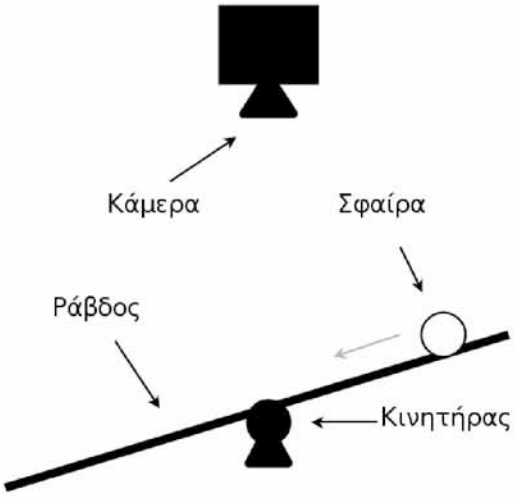
Το σύστημα σφαίρας σε ράβδο, όπως φαίνεται στο παρακάτω σχήμα (Σχήμα 2.4), έχει ως λειτουργία να ισορροπεί μια σφαίρα στο κέντρο της ράβδου, συνήθως με χρήση μιας κάμερας. Το σύστημα αυτό είναι μια απλούστερη μορφή του συστήματος που θα μελετηθεί, είναι και ο λόγος που περιγράφεται εδώ. Σχετικά με το σύστημα σφαίρα σε ράβδο έχουν πραγματοποιηθεί πολλές εργασίες [11, 12, 13], οι οποίες είναι υλικό μελέτης της παρούσας εργασίας.



Σχήμα 2.4: *Παράδειγμα συστήματος σφαίρας σε ράβδο [14]*

Τα κύρια στοιχεία ενός συστήματος σφαίρας σε ράβδο είναι η ράβδος, ο κινητήρας που περιστρέφει τη ράβδο, μια σφαίρα και συνήθως μια κάμερα (Σχήμα 2.5). Στόχος του συστήματος είναι αλλάζοντας την γωνία, να κατευθύνει την σφαίρα στο κέντρο της ράβδου (ή σε οποιοδήποτε άλλο σημείο έχει

οριστεί), το οποίο αποτελεί ένα πρόβλημα ισορροπίας και άρα η σφαίρα επηρεάζεται από πολλές φυσικές δυνάμεις. Για να λυθεί το πρόβλημα πρέπει να υπολογιστεί η συνολική δύναμη που ασκείται στη σφαίρα και να ασκηθεί είτε μια αντίστοιχη, αντίθετη δύναμη με τη βοήθεια της γωνίας της ράβδου, έτσι ώστε να μηδενιστούν οι δυνάμεις που ασκούνται στη σφαίρα, είτε με ελάχιστα μεγαλύτερη ή μικρότερη γωνία για να κατευθυνθεί η σφαίρα προς αντίστοιχη άκρη της ράβδου. Όπως γίνεται αντιληπτό δεν μπορούν να υπολογιστούν με ακρίβεια όλες οι δυνάμεις αυτές, όπως και το ακριβές κέντρο μάζας της σφαίρας με τη χρήση της κάμερας, καθώς και η μη απόλυτη ακρίβεια περιστροφής του κινητήρα, με αποτέλεσμα να υπάρχουν κάποια μικρά ενδεχομένως σφάλματα. Τα σφάλματα αυτά έχουν σαν συνέπεια αποκλίσεις στη γωνία της ράβδου, με τελικό αποτέλεσμα να μην ισορροπεί η σφαίρα. Για να λυθεί αυτό το εμπόδιο λαμβάνεται υπόψη και η προηγούμενη έξοδος (ανατροφοδότηση), υπολογίζεται το σφάλμα και με την βοήθεια του ελεγκτή προσδιορίζεται η καινούρια γωνία που θα εμφανίσει μικρότερη απόκλιση από την επιθυμητή γωνία. Άρα το σύστημα σφαίρας σε ράβδο υλοποιείται κυρίως ως ένα αυτόματο σύστημα ελέγχου κλειστού βρόγχου.

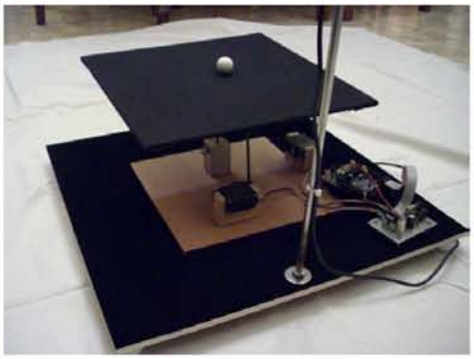


Σχήμα 2.5: *Απλοποιημένο σχήμα του συστήματος σφαίρας σε ράβδο*

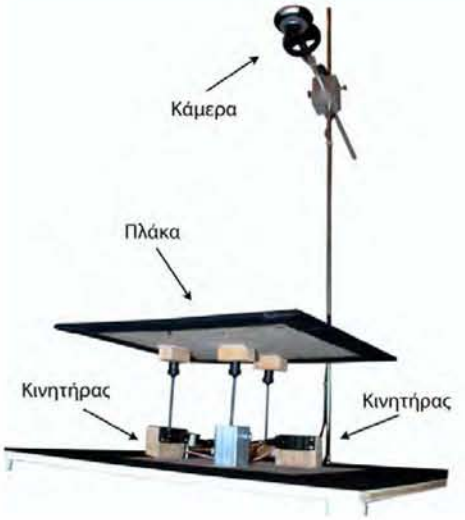
2.3 Περιγραφή του συστήματος σφαίρας σε πλάκα

Το σύστημα σφαίρας σε πλάκα (Σχήμα 2.6) είναι ένα σύστημα σφαίρας σε ράβδο σε δύο διαστάσεις, δηλαδή η επιφάνεια ισορροπίας δεν είναι πια η

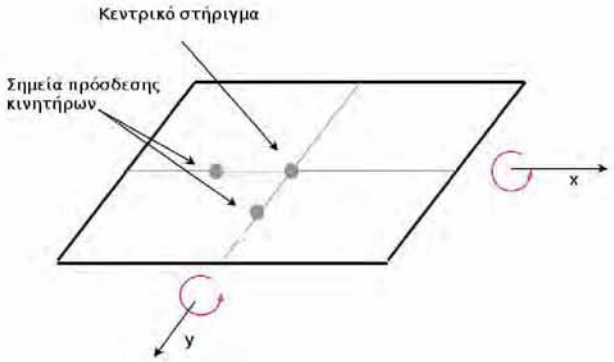
ράβδος (μιας διάστασης) αλλά μια τετράγωνη πλάκα (δύο διαστάσεων) με σημείο επιθυμητής ισορροπίας συνήθως το κέντρο της πλάκας. Εργασίες με θέμα την υλοποίηση ή προσομοίωση του συστήματος σφαίρας σε πλάκα είναι το κύριο υλικό της παρούσας πτυχιακής εργασίας [15, 16, 17].



Σχήμα 2.6: *Ένα παράδειγμα συστήματος σφαίρας σε πλάκα [18]*



Σχήμα 2.7: *Τα κύρια μέρη του συστήματος σφαίρας σε πλάκα [18]*



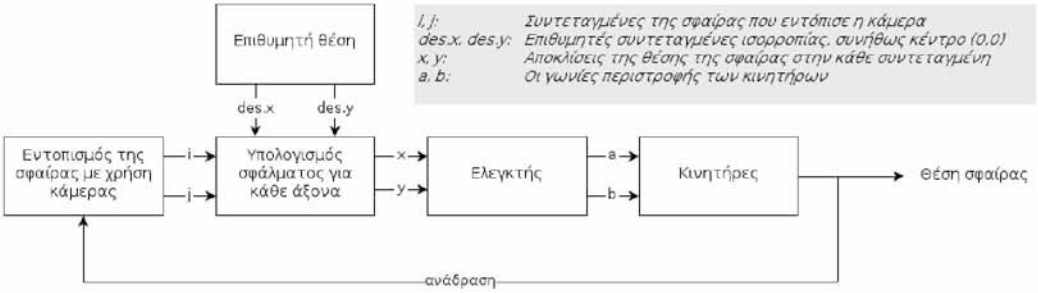
Σχήμα 2.8: *Οι δύο άξονες της πλάκας και τα σημεία πρόσδεσης των κινητήρων*

Το σύστημα λειτουργεί σε δύο διαστάσεις, οπότε χρειάζονται πλέον δύο κινητήρες, έναν για κάθε άξονα (Σχήμα 2.7), άρα είναι δύο βαθμών ελευθερίας. Η πλάκα έχει ένα σταθερό σημείο στο κέντρο της και δύο σημεία σύνδεσης με τους κινητήρες, σε διάταξη ορθής γωνίας (Σχήμα 2.8). Με αυτόν τον τρόπο σύνδεσης επιτυγχάνεται η ανεξάρτητη λειτουργία μεταξύ των αξόνων, οπότε το πρόβλημα μπορεί να λυθεί πια σαν δύο ανεξάρτητα συστήματα σφαίρας

σε ράβδο. Το σύστημα όπως και το σφαίρας-ράβδου υλοποιείται κυρίως σαν σύστημα ελέγχου κλειστού βρόγχου.

Η λειτουργία του συστήματος φαίνεται στο παρακάτω διάγραμμα (Σχήμα 2.9) και συνοπτικά περιγράφεται στα εξής βήματα:

1. Εισάγονται οι επιθυμητές συντεταγμένες, το σημείο στο οποίο θα ισορροπίσει η σφαίρα (συνήθως είναι το κέντρο της πλάκας).
2. Εντοπίζεται η σφαίρα και οι συντεταγμένες της.
3. Υπολογίζονται οι αποκλίσεις της κάθε συντεταγμένη ως προς τις επιθυμητές συντεταγμένες.
4. Ο ελεγκτής λαμβάνει τις αποκλίσεις και προσδιορίζει τη γωνία που πρέπει να εφαρμοστεί στον κάθε κινητήρα.
5. Μετακινείται η σφαίρα και το σύστημα επανέρχεται στο δεύτερο βήμα.



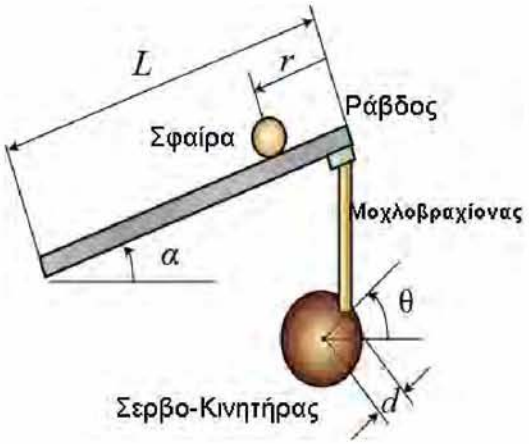
Σχήμα 2.9: *Διάγραμμα του συστήματος σφαίρας σε πλάκα*

2.4 Μαθηματική μοντελοποίηση του συστήματος σφαίρας σε πλάκα

Για να σχεδιαστεί ο ελεγκτής του συστήματος σφαίρας σε πλάκα, πρέπει να αναπτυχθεί ένα μαθηματικό μοντέλο των φυσικών δυνάμεων του συστήματος. Όπως αναφέρθηκε προηγουμένως, το σύστημα μπορεί να διαχωριστεί σε δύο ανεξάρτητα συστήματα σφαίρας σε ράβδο, οπότε αρκεί η λύση ενός συστήματος σφαίρα σε ράβδο.

Η σφαίρα τοποθετείται στη ράβδο, όπου κυλάει κατά μήκος της ράβδου (σε μια διάσταση). Ένας μοχλοβραχίονας συνδέει τη ράβδο με τον κινητήρα (Σχήμα 2.10). Καθώς ο κινητήρας γυρνάει κατά μια γωνία θ, ο μοχλός αναγκάζει τη ράβδο να γυρίσει κατά γωνία α. Όταν η γωνία της ράβδου φύγει

από την οριζόντια θέση, η βαρύτητα προκαλεί το κύλισμα της σφαίρας κατά μήκος της ράβδου. Με τη βοήθεια του ελεγκτή η θέση της σφαίρας στη ράβδο θα καθοδηγείται [19].



Σχήμα 2.10: *Μηχανικό σύστημα σφαίρας σε ράβδο [19]*

Υποθέτουμε ότι δεν έχουμε ολίσθηση της σφαίρας και η τριβή μεταξύ της σφαίρας και της ράβδου είναι αμελητέα. Στο πίνακα 2.1 παρατίθενται οι μεταβλητές και σταθερές του συστήματος.

m	μάζα της σφαίρας
R	ακτίνα της σφαίρας
g	σταθερά βαρύτητας
J	ροπή αδράνειας σφαίρας
r	θέση της σφαίρας πάνω στη ράβδο
L	μήκος της ράβδου (σταθερό σημείο μέχρι το σημείο πρόσδεσης κινητήρα)
α	γωνία της ράβδου
ϑ	γωνία του κινητήρα
d	απόσταση κέντρου κινητήρα και βραχίονα

Πίνακας 2.1: *Μεταβλητές και σταθερές του συστήματος σφαίρας σε ράβδο*

Για να λυθεί το σύστημα χρησιμοποιείται η εξίσωση Lagrangian [20] της κίνησης της σφαίρας [15, 19]:

$$0 = (\frac{J}{R^2} + m)\ddot{r} + mg\sin(\alpha) - mr\dot{\alpha}^2 \tag{2.1}$$

Η γραμμικοποίηση της εξίσωσης ως προς τη γωνία της ράβδου, α=0, μας δίνει τη γραμμική προσέγγιση του συστήματος [19]:

$$(\frac{J}{R^2} + m)\ddot{r} = -mg\alpha \tag{2.2}$$

Η εξίσωση που συνδέει τη γωνία της ράβδου με τη γωνία του κινητήρα μπορεί να προσεγγιστεί από την εξίσωση [19]:

$$\alpha = \frac{d}{L}\theta \tag{2.3}$$

αντικαθιστώντας στην προηγούμενη εξίσωση, έχουμε:

$$(\frac{J}{R^2} + m)\ddot{r} = -mg\frac{d}{L}\theta \tag{2.4}$$

η εξίσωση μετασχηματίζεται με Laplace [19]:

$$(\frac{J}{R^2} + m)R(s)s^2 = -mg\frac{d}{L}\Theta(s) \tag{2.5}$$

Αναδιατάσσοντας, βρίσκουμε τη συνάρτηση μεταφοράς της γωνίας του κινητήρα ($\Theta(s)$) προς τη θέση της σφαίρας ($R(s)$):

$$P(s) = \frac{R(s)}{\Theta(s)} = -\frac{mgd}{L(\frac{J}{R^2} + m)}\frac{1}{s^2} \tag{2.6}$$

Γνωρίζουμε ότι η ροπή αδράνειας για συμπαγή σφαίρα δίνεται από την εξής εξίσωση [21]:

$$J_{Solid} = \frac{2}{5}mR^2 \tag{2.7}$$

και για μη συμπαγή σφαίρα [21]:

$$J_{Hollow} = \frac{2}{3}mR^2 \tag{2.8}$$

Αντίστοιχα η συνάρτηση μεταφοράς (2.6) με τις (2.7), (2.8) μας δίνει τις παρακάτω εξισώσεις:

$$P_{Solid}(s) = \frac{R(s)}{\Theta(s)} = -\frac{5gd}{7L}\frac{1}{s^2} \tag{2.9}$$

$$P_{Hollow}(s) = \frac{R(s)}{\Theta(s)} = -\frac{3gd}{5L}\frac{1}{s^2} \tag{2.10}$$

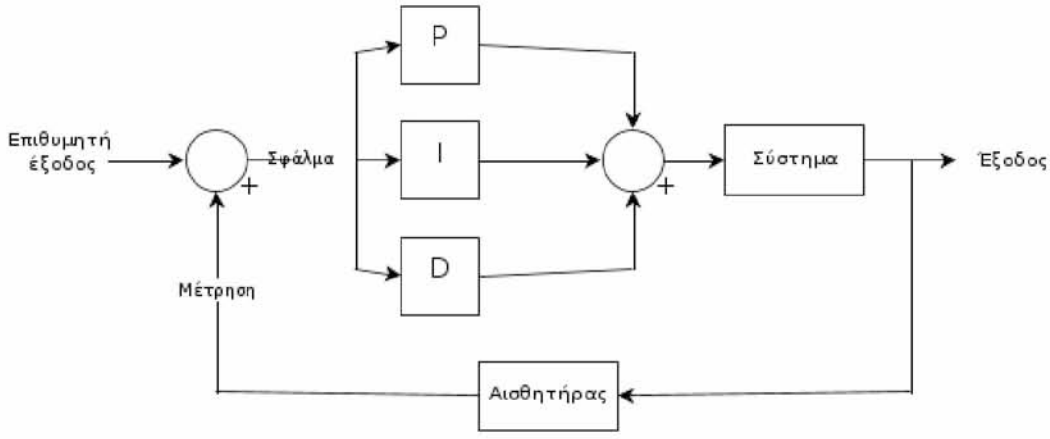
Η συνάρτηση μεταφοράς (2.6) είναι για μια διάσταση. Όταν έχουμε το σύστημα σφαίρας σε πλάκα έχουμε δύο τέτοιες συναρτήσεις, μια για κάθε κινητήρα-άξονα.

Με τις συναρτήσεις αυτές, θεωρητικά, θα μπορούσε να υλοποιηθεί το σύστημα σαν σύστημα ελέγχου ανοιχτού βρόγχου. Στην πραγματικότητα όμως, η κίνηση της σφαίρας επηρεάζεται από πολλές ακόμα (μη μετρίσιμες απόλυτα) δυνάμεις που ασκούνται πάνω της, αλλά και από την μη απόλυτη ακρίβεια του υλικού, δηλαδή της περιστροφής των κινητήρων. Έτσι μπορεί να υλοποιηθεί σαν σύστημα κλειστού βρόγχου, όπου θα παρέχεται επιδιόρθωση του συστήματος, μέσω της ανατροφοδότησης των πραγματικών συνταγμένων της σφαίρας. Στην παρούσα εργασία χρησιμοποιήθηκε ελεγκτής PID, ο οποίος λειτουργεί και χωρίς τη συνάρτηση μεταφοράς.

2.5 PID ελεγκτές και η εφαρμογή τους στο σύστημα σφαίρας σε πλάκα

2.5.1 PID ελεγκτές

Υπάρχουν διάφοροι ελεγκτές αυτόματων συστημάτων, ο πιο διαδεδομένος και πιο εύχρηστος όμως είναι ο PID ελεγκτής (Σχήμα 2.11). Ένας PID ελεγκτής υπολογίζει τη τιμή σφάλματος σαν τη διαφορά μεταξύ της μετρήσιμης τιμής και της επιθυμητής τιμής. Ο σκοπός του ελεγκτή είναι να ελαχιστοποιήσει το σφάλμα μέσα από την ρύθμιση της απόκρισης του συστήματος [22]. Ο ελεγκτής αυτός χρησιμοποιείται σε συστήματα κλειστού βρόγχου.



Σχήμα 2.11: *Αναπαράσταση συστήματος με PID ελεγκτή*

Ο ελεγκτής PID ουσιαστικά είναι ένας αλγόριθμος που υπολογίζει την τιμή του σφάλματος μέσα από τις ξεχωριστές συνιστώσες, και έτσι πολλές φορές ονομάζεται ελεγκτής τριών όρων. Ο όρος PID προκύπτει από τα αρχικά των λέξεων Proportional (αναλογικός), Integral (ολοκλήρωμα) και Deriva-

tive (παράγωγος) που αντιστοιχούν στις τρεις συνιστώσες του, όπου ο όρος P αντιπροσωπεύει το σφάλμα της παρούσας κατάστασης, ο όρος I τη συσσώρευση των σφαλμάτων του παρελθόντος, και ο όρος D είναι μια πρόβλεψη των μελλοντικών σφαλμάτων. Το σταθμισμένο άθροισμα των τριών ορών χρησιμοποιείται για τη ρύθμιση του συστήματος αυτόματου ελέγχου [22]. Η έξοδος του ελεγκτή στο πεδίο του χρόνου περιγράφεται ως εξής [23]:

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau)d\tau + K_D \frac{de(t)}{dt} \tag{2.11}$$

όπου το $e(t)$ είναι η απόκλιση (σφάλμα) τιμής από την επιθυμητή τιμή, που παρατηρείται. Οι μεταβλητές K_P, K_I και K_D είναι απλές σταθερές τιμές, που παίρνουν διάφορες τιμές, για να ρυθμίσουν αναλόγως κάποιο σύστημα, σταθμίζοντας έτσι την κάθε συνιστώσα του ελεγκτή, και ονομάζονται αντίστοιχα:

- K_P = Αναλογικό κέρδος
- K_I = Ολοκληρωτικό κέρδος
- K_D = Διαφορικό κέρδος

Ο αναλογικός όρος παράγει μια τιμή εξόδου, η οποία είναι ανάλογη της τρέχουσας τιμής σφάλματος και ρυθμίζεται από την αναλογική σταθερά κέρδους (K_P). Η χρησιμοποίηση του αναλογικού ελέγχου θα έχει ως αποτέλεσμα τη μείωση, αλλά ποτέ την εξάλειψη, του μόνιμου σφάλματος [22, 23].

Ο ολοκληρωτικός όρος είναι άθροισμα όλων των στιγμιαίων σφαλμάτων από την στιγμή που ξεκίνησε η λειτουργία του συστήματος (ολοκλήρωμα του σφάλματος) και δίνει το συσσωρευμένο σφάλμα που έπρεπε να διορθωθεί προηγουμένως. Το συσσωρευμένο σφάλμα πολλαπλασιάζεται με το ολοκληρωτικό κέρδος (K_I) για τη ρύθμιση της απόκρισης του συστήματος. Η χρησιμοποίηση του ολοκληρωτικού ελέγχου θα εξαλείψει το μόνιμο σφάλμα, αλλά θα χειροτερέψει τη μεταβατική απόκριση [22, 23].

Ο διαφορικός όρος είναι το γινόμενο της μεταβολής του σφάλματος με την πάροδο του χρόνο επί το παραγωγικό / διαφορικό κέρδος (K_D). Ο διαφορικός έλεγχος θα έχει σαν αποτέλεσμα την αύξηση της σταθερότητας του συστήματος και μειώνει το χρόνο απόκρισης [22, 23].

Από τους συνδυασμούς των όρων προκύπτουν διάφορες κατηγορίες PID ελεγκτών (μηδενίζοντας τον ανάλογο όρο):

- P ελεγκτές
- PI ελεγκτές

- PID ελεγκτές
- PD ελεγκτές

2.5.2 Η εφαρμογή PID ελεγκτών στο σύστημα

Το σύοτημα οφαίρας σε πλάκα, όπως αναφέρθηκε προηγουμένως, αποτελείται από δύο κινητήρες και η οφαίρα ισορροπεί σε δύο διαστάσεις, επομένως χρειάζεται έναν ελεγκτή για την κάθε διάοταση. Ο κάθε ελεγκτής θα εξετάζει την απόκλιση της θέοης της οφαίρας από την επιθυμητή θέοη, οτη διάοταση που είναι υπεύθυνος, και θα υπολογίζει την αντίστοιχη γωνία που πρέπει να περιστραφεί ο κινητήρας που ελέγχει, έτοι ώοτε να μειωθεί αυτή η απόκλιοη. Εφόοον η επιθυμητή θέοη είναι το κέντρο της επιφάνειας, για τον έλεγχο της απόκλιοης θα χρηοιμοποιηθεί η απόοταοη της οφαίρας από το κέντρο. Οι όροι του ελεγκτή θα ορίζονται ως:

- K_P* (πραγματική θέοη - επιθυμητή θέοη της οφαίρας)
- K_I* όλες οι οτιγμιαίες αποκλίοεις ουνολικά
- K_D* ταχύτητα της οφαίρας (μεταβολή της απόκλιοης οτο χρόνο)

Στα επόμενα κεφάλαια θα περιγραφεί η διαδικαοία υλοποίηοης του ουοτήματος οφαίρας οε πλάκα με τη βοήθεια του PID ελεγκτή.

Κεφάλαιο 3

Περιγραφή υλικού και λογισμικού που χρησιμοποιήθηκαν για την υλοποίηση του συστήματος

Το σύστημα κατασκευάστηκε με το πακέτο ρομποτικών κατασκευών Lego Mindstorms NXT2.0 και ένα κινητό smartphone. Το πρόγραμμα του συστήματος τρέχει αποκλειστικά μόνο στο smartphone με λειτουργικό android. Η γλώσσα προγραμματισμού, που χρησιμοποιήθηκε είναι η Java. Συγκεκριμένα όλη η φυσική κατασκευή αποτελείται από "τουβλάκια", το IntelligentBrick και τους κινητήρες της Lego Mindstorms, μια επίπεδη επιφάνεια, μια κόκκινη οφαίρα και το κινητό smartphone. Η επικοινωνία του κινητού με το IntelligentBrick διεκπεραιώνεται μέσω της τεχνολογίας bluetooth.

Το Lego Mindstorms NXT2.0 επιλέχθηκε για το μικρό κόστος και την ευκολία συναρμολόγησης του συστήματος. Το κινητό smartphone επιλέχθηκε για να προσθέσει επιπλέον αισθητήρες στο σύστημα (συγκεκριμένα την κάμερα) και για να γίνει το σύστημα ανεξάρτητο, αφού η κινητή συσκευή είναι αρκετά ισχυρή επεξεργαστική μονάδα, ώστε να μην χρειάζεται τη χρήση ενός υπολογιστή. Αυτό έχει σαν αποτέλεσμα το σύστημα να είναι φορητό.

3.1 Υλικό

3.1.1 Lego Mindstorms NXT2.0

Το Lego Mindstorms είναι ένα σύνολο κομματιών συναρμολόγησης ρομπότ που παράγει η εταιρία Lego. Το σύνολο αυτό αποτελείται από προγραμμα-

τιζόμενα brick, σέρβο-κινητήρες, αισθητήρες και διάφορα τεχνικά κομμάτια Lego κατάλληλα για να χτίσει ο χρήστης διάφορα μηχανικά συστήματα (Σχήμα 3.1). Αυτά τα κομμάτια είναι συμβατά με όλες τις σειρές της Lego [24].

Σχήμα 3.1: *Ένα ανδροειδές ρομπότ από το σετ NXT [25]*

Οι πιο γνωστές εκδόσεις Lego Mindstorms, με σειρά παραγωγής, είναι [24]:

- Lego Mindstorms RCX
- Lego Mindstorms NXT
- Lego Mindstorms NXT2.0
- Lego Mindstorms EV3 (Αναμένεται να κυκλοφορήσει το φθινόπωρο του 2013)

Η πιο πρόσφατη έκδοση στην αγορά είναι η Lego Mindstorms NXT2.0. Κυκλοφόρησε στις 5 Αυγούστου του 2009 και αποτελείται από 619 κομμάτια (Σχήμα 3.2) [24]. Μέσα στο σύνολο αυτό βρίσκεται το IntelligentBrick,

τρεις σερβο-κινητήρες και τέσσερεις αισθητήρες (δύο αισθητήρες αφής, ένας αισθητήρας υπερήχων και ένας αισθητήρας φωτός και χρώματος).

Σχήμα 3.2: *Τα κομμάτια του NXT2.0 [26]*

Το IntelligentBrick είναι ένα ευφυές ελεγχόμενο από τον υπολογιστή brick. Μπορεί να προγραμματιστεί σε διάφορες γλώσσες, οι πιο γνωστές είναι brickOS (C/C++), LabVIEW, leJOS (Java), Robotc (C), MatLAB [24]. Αποτελείται από μια ασπρόμαυρη οθόνη και τέσσερα κουμπιά πλοήγησης στο μενού του brick, επίσης έχει τέσσερεις εισόδους (θύρες 1, 2, 3, 4) για τους αισθητήρες και τρεις εξόδους (θύρες A, B, C) για τους σερβο-κινητήρες (Σχήμα 3.3). Συνδέεται στον υπολογιστή είτε με καλώδιο USB, είτε με ασύρματη επικοινωνία bluetooth. Λειτουργεί με 6 μπαταρίες AA. Τα τεχνικά χαρακτηριστικά του brick βρίσκονται στο παράρτημα Δ'.

Οι σερβο-κινητήρες προσφέρουν στο ρομπότ τη δυνατότητα της κίνησης. Επίσης κάθε σερβο-κινητήρας έχει ενσωματωμένο αισθητήρα περιστροφής. Ο αισθητήρας περιστροφής διαβάζει τις μοίρες της περιστροφής του κινητήρα με ακρίβεια $+/-$ μια μοίρα. Με τη βοήθεια του αισθητήρα μπορεί να ρυθμιστεί η ταχύτητα του κινητήρα και το πόσες μοίρες θα περιστραφεί [28].

Οι αισθητήρες προσφέρουν στο ρομπότ δυνατότητα αντίληψης του περιβάλλοντος γύρω του. Συγκεκριμένα ο αισθητήρας αφής αντιλαμβάνεται πότε πιέζεται από κάτι και πότε απελευθερώνεται. Ο αισθητήρας υπερήχων δίνει στο ρομπότ τη δυνατότητα να βλέπει, να αναγνωρίζει αντικείμενα, να μετρά-

Σχήμα 3.3: *Το brick με συνδεδεμένους κινητήρες και αισθητήρες [27]*

ει την απόσταση και να ανιχνεύει κίνηση με υπέρηχους. Η απόσταση που μπορεί να μετρήσει είναι από 0 έως 255 εκατοστά με ακρίβεια $+/-$ τρία εκατοστά. Τέλος ο αισθητήρας φωτός και χρώματος προσφέρει επίσης κάποια μορφή όρασης στο ρομπότ. Ο αισθητήρας φωτός και χρώματος έχει τη δυνατότητα να ανιχνεύει 6 διαφορετικά χρώματα, να αντιλαμβάνεται την ένταση του φωτός στο χώρο, την ένταση κάποιου χρώματος σε μια επιφάνεια. Επιπλέον λειτουργεί και σαν λάμπα με 3 διαφορετικά χρώματα [28].

3.1.2 Android Smartphone

Το πρώτο κινητό smartphone με λειτουργικό Android ήταν το HTC Dream και κυκλοφόρησε για πρώτη φορά στις 22 Οκτωβρίου του 2008. Στις μέρες μας, τα κινητά Android έχουν φτάσει σε ρυθμό πωλήσεων 1.3 εκατομμυρίων συσκευών την ημέρα και συνολικά 500 εκατομμύρια ενεργοποιημένες συσκευές σε όλο τον κόσμο [29]. Ο λόγος που ξεχωρίζει το κινητό Android είναι το ειδικά σχεδιασμένο λειτουργικό Android που μπορεί να υποστηριχθεί από διάφορες κινητές συσκευές που πληρούν τις προδιαγραφές σε συνδυασμό με μια πληθώρα χαρακτηριστικών που απαιτούνται για την πλήρη εκμετάλλευση των δυνατοτήτων που προσφέρει το συγκεκριμένο λειτουργικό. Μερικά από τα χαρακτηριστικά των κινητών Android (όπως και άλλων smartphones) είναι τα εξής [30, 31]:

- Αρκετά γρήγοροι και μικροί επεξεργαστές
- Εσωτερική και εξωτερική (επεκτάσιμη) μνήμη
- Οθόνη πολλαπλής αφής
- Πολυεπεξεργασία (Multitasking)
- Camera
- Συνδεσιμότητα, WiFi και Bluetooth
- και αισθητήρες όπως ψηφιακή πυξίδα, GPS, επιταχυνσιόμετρο, γυροσκόπιο, βαρόμετρο, μαγνητόμετρο, θερμόμετρο, αισθητήρας πίεσης κτλ..

Τα τεχνικά χαρακτηριστικά της κινητής συσκευής που χρησιμοποιήθηκε στην εργασία βρίσκονται στο παράρτημα Ε'.

3.2 Λογισμικό

Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε είναι η Java, με βιβλιοθήκες του Android SDK και του OpenCV, και οι εντολές Direct commands του NXT. Το περιβάλλον ανάπτυξης, όπου υλοποιήθηκε το πρόγραμμα του συστήματος, είναι το Eclipse.

3.2.1 Λειτουργικό σύστημα Android

Το Android είναι ένα λειτουργικό σύστημα για συσκευές κινητής τηλεφωνίας, το οποίο τρέχει τον πυρήνα Linux. Αρχικά αναπτύχθηκε από την Android, Inc. που χρηματοδοτήθηκε και αργότερα, το 2005, εξαγοράστηκε από την Google. Η πρώτη παρουσίαση της πλατφόρμας Android έγινε στις 5 Νοεμβρίου 2007, παράλληλα με την ανακοίνωση της ίδρυσης του οργανισμού Open Handset Alliance, μιας κοινοπραξίας 48 τηλεπικοινωνιακών εταιριών, εταιριών ανάπτυξης λογισμικού καθώς και κατασκευής hardware, οι οποίες είναι αφιερωμένες στην ανάπτυξη και εξέλιξη ανοιχτών προτύπων στις συσκευές κινητής τηλεφωνίας. Η ανάπτυξη του λειτουργικού συνεχίστηκε από την Google και τον οργανισμό Open Handset Alliance με την κυκλοφορία αρκετών εκδόσεων του λειτουργικού. Στο πίνακα 3.1 παρουσιάζονται οι εκδόσεις και οι ημερομηνίες κυκλοφορίας τους και στο σχήμα 3.4 εικονίζονται το λογότυπο του λειτουργικού και τα λογότυπα των εκδόσεων. Για τις κωδικές ονομασίες,

API level	Έκδοση	Κωδική ονομασία	Ημερ/νία Κυκλοφορίας
1	1.0	alpha	23 Σεπτεμβρίου 2008
2	1.1	beta	9 Φεβρουαρίου 2009
3	1.5	Cupcake	30 Απριλίου 2009
4	1.6	Donut	15 Σεπτεμβρίου 2009
5, 6, 7	2.0 - 2.1	Eclair	26 Οκτωβρίου 2009
8	2.2 - 2.2.3	Froyo	20 Μαΐου 2010
9, 10	2.3 - 2.3.7	Gingerbread	6 Δεκεμβρίου 2010
11, 12, 13	3.0 - 3.2	Honeycomb	22 Φεβρουαρίου 2011
14, 15	4.0 - 4.0.4	Ice Cream Sandwich	19 Οκτωβρίου 2011
16, 17	4.1 - 4.2	Jelly Beam	27 Ιουνίου 2012
18	5.0	Key Lime Pie	Αναμένεται να κυκλοφορήσει

Πίνακας 3.1: *Εκδόσεις του λειτουργικού Android [33]*

Σχήμα 3.4: *Το λογότυπο του λειτουργικού Android και των εκδόσεών του [34]*

ή απλώς ονόματα των εκδόσεων, επιλέχθηκαν ονόματα διαφόρων γλυκών σε αλφαβητική ακολουθία [29, 32].

Το λειτουργικό Android είναι ανοιχτού κώδικα λογισμικό δημοσιευμένο υπό την άδεια Apache. Το λειτουργικό αυτό έχει προσελκύσει πολλούς developers οι οποίοι αναπτύξανε πάνω από 700.000 εφαρμογές (apps - applications) για Android, ο λόγος είναι η ευκολία ανάπτυξης της εφαρμογής (χρησιμοποιείται κυρίως η γλώσσα προγραμματισμού Java) και η δυνατότητα εκτέλεσης της εφαρμογής σε πληθώρα διαφορετικών συσκευών κινητής τηλεφωνίας, με διαφορετικά τεχνικά χαρακτηριστικά και διαφορετικούς κατασκευαστές της συσκευής, αρκεί να τρέχει λειτουργικό Android [29]. Στις μέρες μας το λειτουργικό Android έχει επικρατήσει και μπορεί να βρεθεί πια όχι μόνο σε συσκευές κινητής τηλεφωνίας αλλά και σε φορητούς υπολογιστές (tablets και netbooks), ακόμα και σε διάφορες συσκευές όπως ρολόγια, ψυγεία, τηλεοράσεις, φωτογραφικές μηχανές κτλ.

3.2.2 Ανάπτυξη εφαρμογής για το λειτουργικό Android

Η ανάπτυξη της πρώτης εφαρμογής Android ξεκινάει με την εγκατάσταση των εργαλείων, όπου τα σημαντικότερα είναι:

- Java Developer Kit (JDK)
- Περιβάλλον ανάπτυξης: μπορεί να είναι ένας απλός κειμενογράφος, όπως το Notepad, ή ένα πλήρες περιβάλλον ανάπτυξης όπως το Eclipse.
- Android software development kit (SDK): συμπεριλαμβάνει ένα σύνολο εργαλείων όπως βιβλιοθήκες, τεκμηρίωση, δείγματα κώδικα, debugger, tutorials, emulator.

Αφού εγκαταστασταθούν τα απαραίτητα εργαλεία μπορεί να αρχίσει η ανάπτυξη της εφαρμογής. Η γλώσσα προγραμματισμού είναι η Java μαζί με τις βιβλιοθήκες Android που παρέχει το Android SDK.

Ένα σημαντικό εργαλείο που παρέχει το SDK είναι τα emulators. Τα emulators δίνουν την δυνατότητα δημιουργίας εικονικών κινητών συσκευών (προσομοίωση λειτουργίας σε αντίστοιχες συσκευές) έτσι ώστε ο developer να μπορεί να τρέξει την εφαρμογή χωρίς να είναι υποχρεωμένος να έχει κάποια συσκευή Android, αλλά και να δει πως λειτουργεί η εφαρμογή του σε συσκευές διαφορετικών διαστάσεων και με διαφορετικές αναλύσεις και αναλογίες οθόνης. Παράδειγμα προσομοίωσης μια κινητής συσκευής με Android 4.1 φαίνεται στο σχήμα 3.5.

Τέλος ο προγραμματιστής μπορεί να δημοσιεύσει την εφαρμογή του στο Google Play από όπου μπορεί να το κατεβάσει και να το εγκαταστήσει στη συσκευή του οποιοσδήποτε χρήστης Android.

Σχήμα 3.5: *Εικονική συσκευή με Android 4.1*

3.2.3 Βιβλιοθήκη OpenCV

Η OpenCV (Open Source Computer Vision Library) είναι μια βιβλιοθήκη συναρτήσεων με κύριο στόχο την επεξεργασία βίντεο πραγματικού χρόνου. Αναπτύχθηκε από την Intel και προσφέρεται δωρεάν ως ελεύθερο λογιομικό [35]. Η βιβλιοθήκη έχει περισσότερο από 2500 βελτιστοποιημένους αλγορίθμους, στους οποίους περιλαμβάνεται μια ολοκληρωμένη σειρά αλγορίθμων τεχνητής όρασης και εκμάθησης μηχανών. Οι αλγόριθμοι αυτοί μπορούν να χρησιμοποιηθούν για [36]:

- Ανίχνευση και αναγνώριση προσώπων
- Προσδιορισμός αντικειμένων
- Ταξινόμηση ανθρώπινων ενεργειών σε βίντεο
- Παρακολούθηση κινούμενων αντικειμένων

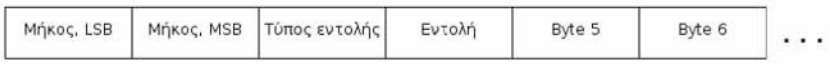
- Αναζήτηση παρόμοιων εικόνων σε μια βάση εικόνων
- Αφαίρεση κόκκινων ματιών από φωτογραφίες κτλ.

Η βιβλιοθήκη αρχικά ήταν γραμμένη σε γλώσσα C και είχε διασυνδέσεις με γλώσσες όπως C#, Python, Ruby και Java. Αργότερα γράφτηκε σε γλώσσα C++, απλουστεύοντας τις γραμμές κώδικα (δέσμευση χώρου κτλ.), αλλά έγινε πιο δύσκολη στην παροχή διασυνδέσεων σε άλλες γλώσσες. Τελευταία αναπτύσσεται με γοργούς ρυθμούς η πλήρης διασύνδεση της βιβλιοθήκης με τις κάρτες γραφικών CUDA[36].

Η βιβλιοθήκη προσφέρει διασύνδεση σε Android (OpenCV4Android) η οποία θα χρησιμοποιηθεί για εντοπισμό της σφαίρας πάνω στην επίπεδη επιφάνεια.

3.2.4 Lego Mindstorms Direct Commands

Το NXT brick να μπορεί να δεχτεί εντολές από συσκευές συνδεδεμένες, μέσω bluetooth, μαζί του. Οι εντολές αυτές λέγονται Direct commands και ερμηνεύονται από το NXT brick σε συγκεκριμένες ενέργειες χωρίς την απαίτηση να τρέχει κάποιος κώδικας σε αυτό. Με τις εντολές αυτές είναι εφικτό το ρομπότ να ελέγχεται από κάποια φορητή συσκευή όπως ένα κινητό τηλέφωνο. Οι εντολές έχουν τη μορφή αλληλουχίας byte που θα σταλούν μέσω του bluetooth στο brick από το οποίο θα επιστρέψει και η κατάλληλη απάντηση πάλι σε μορφή αλληλουχίας byte. Το σύνολο των εντολών Direct commands ονομάζεται πρωτόκολλο επικοινωνίας του Mindstorms NXT [37].



Σχήμα 3.6: *Γενική αρχιτεκτονική του πρωτόκολλου επικοινωνίας*

Στο σχήμα 3.6 παρουσιάζεται η γενική αρχιτεκτονική του πρωτοκόλλου επικοινωνίας. Τα πρώτα δύο byte δηλώνουν το μήκος της εντολής. Το τρίτο byte δηλώνει τον τύπο της εντολής. Το τέταρτο byte δηλώνει την εντολή και τα υπόλοιπα byte είναι τα στοιχεία της εντολής. Το κάθε πακέτο εντολής μπορεί να έχει μήκος μέχρι 64 byte χωρίς να περιλαμβάνονται σε αυτά τα δύο πρώτα byte.

Ένα απλό παράδειγμα [38], εντολή να παίξει έναν τόνο ήχου (χωρίς τα byte μήκους):

```
Byte 0: 0x00
Byte 1: 0x03
Byte 2 - 3: Frequency for the tone, 200 - 14000 Hz
Byte 3 - 5: Duration of the tone, ms
```

όπου στο πρώτο byte δηλώνεται ο τύπος της εντολής και συγκεκριμένα το $0x00$ σημαίνει ότι είναι απαραίτητη η απάντηση του brick. Στο δεύτερο byte στέλνεται ο κωδικός της εντολής, το $0x03$ για να παίξει ένα τόνο. Στα byte 2 - 5 δηλώνονται η συχνότητα και η διάρκεια του ήχου. Η απάντηση από τη συγκεκριμένη εντολή είναι η παρακάτω:

```
Byte 0: 0x02
Byte 1: 0x03
Byte 2: Status Byte
```

Το πρώτο byte είναι ο τύπος της εντολής όπου το $0x02$ σημαίνει επιστρεφόμενο μήνυμα, το δεύτερο byte δηλώνει των κωδικό της εντολής για την οποία ήρθε η απάντηση και το τρίτο byte παρουσιάζει την κατάσταση της εντολής, αν εκτελέστηκε επιτυχώς ή το σφάλμα που προέκυψε. Με το ίδιο σκεπτικό πραγματοποιούνται και οι υπόλοιπες εντολές. Η παρουσίαση και εξήγηση των εντολών παρουσιάζεται στα εγχειρίδια της Lego [37, 38].

3.2.5 Eclipse Editor

Το Eclipse είναι ένα περιβάλλον ανάπτυξης λογισμικού σε πολλές γλώσσες προγραμματισμού και περιλαμβάνει ένα ολοκληρωμένο περιβάλλον ανάπτυξης (integrated development environment - IDE) και είναι εύκολα επεκτάσιμο με plugin. Είναι γραμμένο κυρίως σε Java. Υποστηρίζει μεγάλο σύνολο γλωσσών όπως Java, C, C++, Fortran, Perl, PHP, Python, Ruby, Android [39]. Το περιβάλλον ανάπτυξης μαζί με το ADT plugin (Android Developer Tools) είναι πολύ χρήσιμα εργαλεία ανάπτυξης Android εφαρμογών. Συγκεκριμένα το ADT plugin προοφέρει [40]:

- Εύκολη δημιουργία projects
- Αποσφαλμάτωση εφαρμογών
- Εξαγωγή .apk αρχείων
- Γενικά γρήγορη ανάπτυξη εφαρμογών

Τέλος χρησιμοποιήθηκαν επίσης οι εφαρμογές: Photoshop, Kile, LEGO Digital Design 4.3, draw.io (http://www.diagram.ly/).

Κεφάλαιο 4

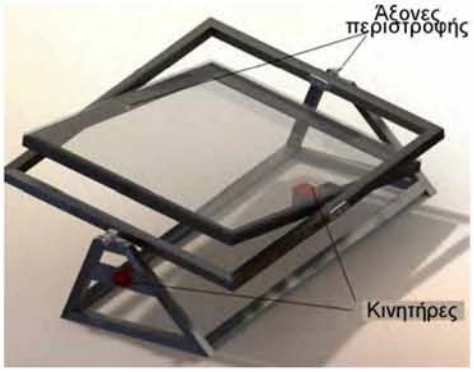
Υλοποίηση του συστήματος σφαίρας σε πλάκα

4.1 Κατασκευή του συστήματος

Η υλοποίηση του συστήματος ξεκινά με τη σχεδίαση της κατασκευής και τη συναρμολόγησή της. Για την κατασκευή του συστήματος αναλύθηκαν διάφορα ολοκληρωμένα συστήματα σφαίρας σε πλάκα με διαφορετικούς τρόπους κατασκευής, οι πιο διαδεδομένοι τρόποι κατασκευής φαίνονται στα παρακάνω σχήματα (Σχήμα 4.1, Σχήμα 4.2). Με γνώμονα το περιορισμένο και συγκεκριμένο σύνολο κομματιών Lego επιλέχθηκε ο τρόπος κατασκευής του σχήματος 4.2.

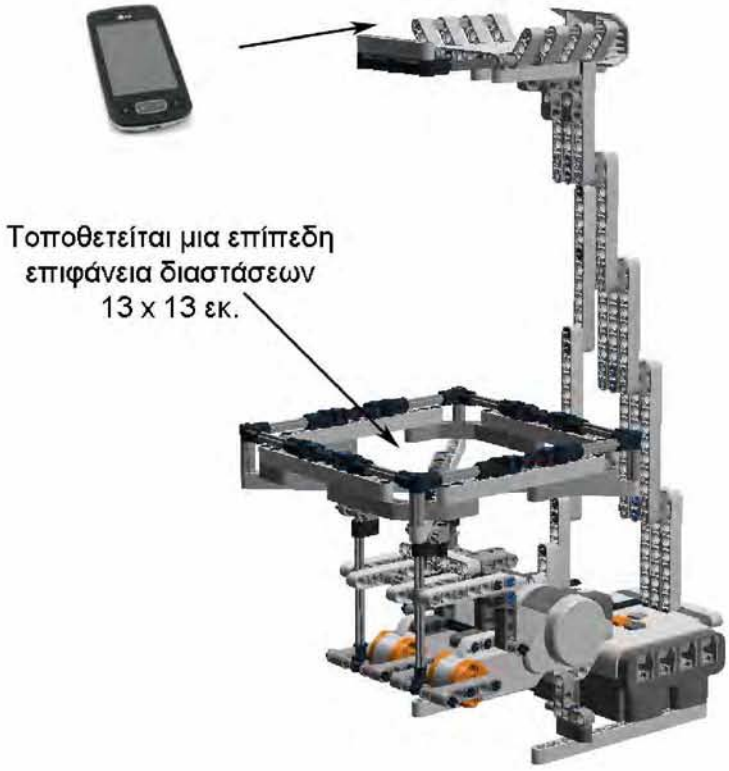
Τα σημαντικότερα σημεία της κατασκευής είναι:

- Η επίπεδη επιφάνεια (πλάκα) και η στήριξή της.
- Οι δύο κινητήρες, η στήριξή τους και η σύνδεσή τους με την επίπεδη επιφάνεια.
- Η στήριξη του κινητού.
- Η σταθερότητα της κατασκευής.



Σχήμα 4.1: *Κατασκευή του συστήματος με πλαίσια [15]*

Σχήμα 4.2: *Κατασκευή του συστήματος με άξονες [18]*



Σχήμα 4.3: *Τελική μορφή της κατασκευής*

Οι απαιτήσεις της κατασκευής αναλύονται στα παρακάτω:

- Η επίπεδη επιφάνεια πρέπει να έχει σταθερό σημείο στήριξης στο κέντρο της και να συνδέεται με τους κινητήρες με τέτοιο τρόπο ώστε τα σημεία σύνδεσης σε σχέση με το κέντρο να σχηματίζουν ορθή γωνία. Οι συνδέσεις στους κινητήρες και στο κέντρο πρέπει να είναι με τέτοιο τρόπο ώστε να διευκολύνουν την ομαλή αλλαγή γωνίας της πλάκας προς όλες τις πλευρές, τουλάχιστον κατά 10 μοίρες.

- Οι κινητήρες πρέπει να συνδεθούν σταθερά και με την περιστροφική κίνηση τους να περιστρέφουν την πλάκα. Οι άξονες περιστροφής των κινητήρων πρέπει να είναι κάθετοι μεταξύ τους.

- Το κινητό πρέπει να βρίσκεται πάνω από την επίπεδη επιφάνεια, σε οριζόντια θέση, με την κάμερα του κινητού να κοιτάει προς την επίπεδη επιφάνεια και σε τέτοιο ύψος ώστε η κάμερα του κινητού να "βλέπει" όλη την επιφάνεια.

- Τέλος όλη η κατασκευή πρέπει να ισορροπεί και να έχει σταθερότητα κατά τη διάρκεια λειτουργίας των κινητήρων.

Λαμβάνοντας υπόψη τα κύρια σημεία με τις απαιτήσεις τους και μετά από αρκετούς πειραματισμούς η κατασκευή κατέληξε στην τελική μορφή της (Σχήμα 4.3). Αποτελείται από 309 κομμάτια της Lego Mindstorms, μια επίπεδη επιφάνεια (13x13 εκ.) και ένα κινητό. Οι οδηγίες συναρμολόγησης της κατασκευής δημιουργήθηκαν με το πρόγραμμα Lego Digital Design 4.3 και βρίσκονται στο παράρτημα Α'.

4.2 Προγραμματισμός του συστήματος

Για να λειτουργήσει το σύστημα ήταν απαραίτητη η συγγραφή κώδικα. Ο κώδικας είναι εξολοκλήρου γραμμένος σε Java, Android βιβλιοθήκες, OpenCV βιβλιοθήκες και Lego Direct Commands (τα οποία στην ουσία είναι αλληλουχίες με bytes), έχει τη μορφή μιας εφαρμογής (app) Android και εκτελείται αποκλειστικά στην κινητή συσκευή. Το NXT brick, δεν χρειάζεται να εκτελέσει κάποιο κώδικα, συνεπώς ούτε να προγραμματιστεί, γιατί τα Direct Commands ερμηνεύονται απευθείας από το Firmware του NXT brick.

Τα βήματα του αλγορίθμου της εφαρμογής συνοπτικά είναι τα εξής (Σχήμα 4.4):

1. Εκκίνηση της εφαρμογής, εντοπισμός του NXT brick και εδραίωση επικοινωνίας με αυτό.

2. Εκκίνηση της κάμερας και των αλγορίθμων εντοπισμού της σφαίρας (βιβλιοθήκες OpenCV).

3. Εντοπισμός της σφαίρας και σύγκριση της θέσης της με την επιθυμητή θέση. Αν είναι στην επιθυμητή θέση, το βήμα επαναλαμβάνεται μέχρι η σφαίρα να φύγει από την επιθυμητή θέση.

4. Εφόσον δεν είναι στην επιθυμητή θέση η σφαίρα, γίνεται υπολογισμός με την βοήθεια του PID ελεγκτή των γωνιών που πρέπει να περιστραφούν οι κινητήρες.

5. Αποστέλλεται η εντολή στο NXT brick για περιστροφή των κινητήρων και ο αλγόριθμος επιστρέφει στο βήμα 3.

4.2.1 Η δομή μιας εφαρμογής Android

Προτού περιγραφεί η ανάπτυξη του κώδικα της εφαρμογής, πρέπει να επεξηγηθεί και να περιγραφεί η δομή μιας εφαρμογής Android.

Οι εφαρμογές Android είναι γραμμένες σε γλώσσα προγραμματισμού Java και βιβλιοθήκες που βρίσκονται στο Android SDK. Ο πηγαίος κώδικας, μαζί με όλα τα δεδομένα και τα αρχεία μιας εφαρμογής μεταγλωττίζονται από το αντίστοιχο εργαλείο του Android SDK με τελικό προϊόν ένα αρχείο με κατάληξη .apk (android package). Το αρχείο αυτό μπορεί να εγκατασταθεί σε συσκευές με λειτουργικό Android.

Κάθε εφαρμογή Android απαρτίζεται από ένα ή περισσότερα δομικά στοιχεία. Τα δομικά αυτά στοιχεία χωρίζονται σε τέσσερα είδη, ανάλογα με το σκοπό και το κύκλο ζωής τους. Τα τέσσερα είδη είναι τα εξής [41]:

Σχήμα 4.4: *Συνοπτικός αλγόριθμος της εφαρμογής για το σύστημα σφαίρας σε πλάκα*

Activities:

Κάθε activity (δραστηριότητα) αντιπροσωπεύει μια οθόνη (ή παράθυρο) με τη διεπαφή του χρήστη. Τα στοιχεία για τη διεπαφή συνήθως τα τραβάει από κάποιο αρχείο xml με τη συνάρτηση setContentView(). Παράδειγμα, μια εφαρμογή ανάγνωσης ενημερωτικών άρθρων (όπως το Google Reader) έχει ένα activity με τους τίτλους των άρθρων σε μια λίστα και ένα άλλο activity που εμφανίζει ολόκληρο ένα συγκεκριμένο άρθρο για ανάγνωση. Το activity είναι το πιο κοινό συστατικό και συναντάτε σχεδόν σε όλες τις εφαρμογές.

Services:

Ένα service (υπηρεσία) είναι το κομμάτι του κώδικα που τρέχει στο παρασκήνιο και δεν έχει κάποια διεπαφή με το χρήστη. Παράδειγμα, ένα service μπορεί να παίζει μουσική ή να κατεβάζει δεδομένα από το διαδίκτυο χωρίς να αλληλεπιδράει με το χρήστη.

Content providers:

Ένας content provider (πάροχος περιεχομένου) διαχειρίζεται ένα σύνολο δεδομένων της εφαρμογής. Παρέχει τη δυνατότητα να χρησιμοποιηθούν δεδομένα από άλλες εφαρμογές ή να διαμοιραστούν τα δεδομένα στις άλλες εφαρμογές.

Broadcast receivers:

Ο broadcast receiver (δέκτης εκπομπών-μεταδόσεων) είναι το δομικό στοιχείο που ανταποκρίνεται στις ειδοποιήσεις του συστήματος όπως παράδειγμα χαμηλή στάθμη μπαταρίας.

Για να μπορέσει το σύστημα να ξεκινήσει κάποιο συστατικό της εφαρμογής πρέπει να επιβεβαιωθεί ότι υπάρχει αυτό το συστατικό "διαβάζοντας" το αρχείο manifest (AndroidManifest.xml) της εφαρμογής. Κατά συνέπεια όλα τα δομικά στοιχεία της εφαρμογής πρέπει να δηλώνονται στο αρχείο manifest, το οποίο προσδιορίζει επίσης τα δικαιώματα, το ελάχιστο API level, τις βιβλιοθήκες API (π.χ. Google Maps library) της εφαρμογής κ.ά [41].

Πρέπει να σημειωθεί ότι ο κύκλος ζωής κάθε δομικού στοιχείου συνοδεύετε από κάποιες συγκεκριμένες μεθόδους όπως φαίνεται παράδειγμα στο σχήμα 4.5 που παριστάνει το κύκλο ζωής ενός activity.

Τέλος, ο προγραμματισμός μιας εφαρμογής ξεκινάει με την δημιουργία ενός project, το οποίο με λίγα λόγια είναι ένα σύνολο αρχείων της εφαρμογής (Σχήμα 4.6). Στο eclipse το project δημιουργείται πολύ εύκολα με το File → New → Android Application Project που παράγει κάποια αρχικοποιημένα αρχεία όπως το αρχείο manifest, ένα αρχείο xml για τη διάταξη του γραφικού

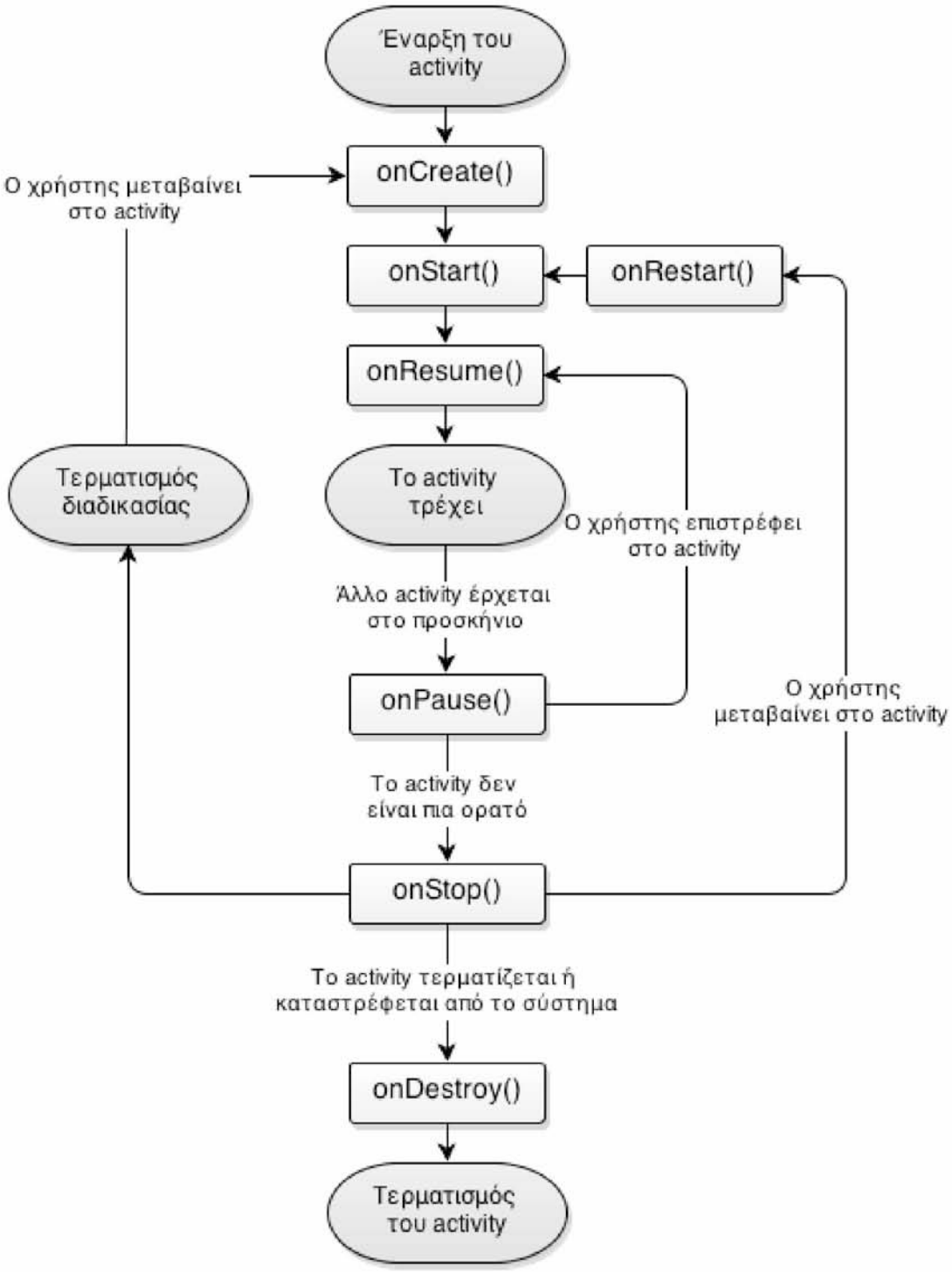
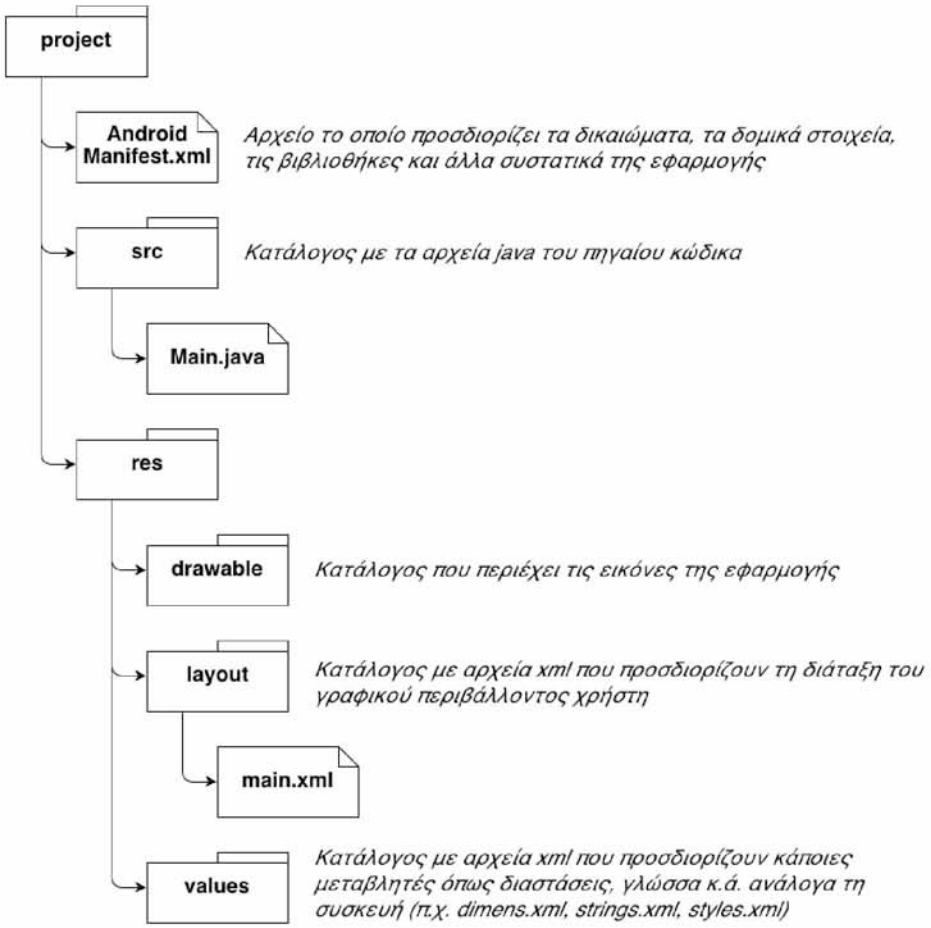


Figure 4.5: *Κύκλος ζωής ενός activity μιας εφαρμογής [41]*

περιβάλλοντος χρήστη του αρχικού activity, και ένα αρχείο Java με υλοποιημένη τη μέθοδο onCreate() όπου καλεί με τη με τη μέθοδο setContentView() τη διάταξη του γραφικού περιβάλλοντος χρήστη. Αυτό το αρχικοποιημένο project συνήθως παράγει μια εφαρμογή η οποία όταν τρέξει θα εμφανίσει το κλασικό κείμενο "Hello world".



Σχήμα 4.6: *Οι σημαντικότεροι κατάλογοι και αρχεία ενός Android project*

4.2.2 Καθιέρωση επικοινωνίας του NXT με το κινητό Android

Το πρώτο σημαντικό μέρος της εφαρμογής είναι η επίτευξη της επικοινωνίας ανάμεσα στη κινητή συσκευή και στο NXT brick. Έτσι μετά τη δημιουργία του project, γράφτηκε κώδικας για την επικοινωνία της συσκευής με το brick. Για λόγους απλούστευσης της εφαρμογής, πήραμε ως δεδομένο ότι το bluetooth

και στις δύο συσκευές είναι ενεργοποιημένο και έχει προηγηθεί το pairing[1] των συσκευών.

Έτσι μέθοδος για τον εντοπιομό του brick είναι [41, 42]:

```
1  public void findRobot(){
2    try{
3      // Get default bluetooth adapter
4      btAdapter = BluetoothAdapter.getDefaultAdapter();
5      // Get paired bluetooth devices
6      pairedDevices = btAdapter.getBondedDevices();
7      // set iterator over a sequence of bluetooth devices, like a list
8      Iterator<BluetoothDevice> it = pairedDevices.iterator();
9      // Scan the iterator/list for specific bluetooth device,
10     // the robot and call function to connect
11     while (it.hasNext()){
12       BluetoothDevice bd = it.next();
13       if (bd.getName().equalsIgnoreCase("NXT")) {
14         connectToRobot(bd);
15         return;
16       }
17     }
18   }catch (Exception e){
19     Log.e(TAG,"Failed in findRobot() [" + e.getMessage() +"]");
20   }
21 }
```

όπου καταλήγει στις γραμμές 11-17 να αναζητά το brick (με το όνομα NXT) ανάμεσα στις συσκευές σε ζεύξη με τη συσκευή που τρέχει την εφαρμογή, και αφού το εντοπίσει καλεί τη μέθοδο connectToRobot(), η οποία υλοποιείται ως εξής [41, 42]:

```
1  private void connectToRobot(BluetoothDevice bd){
2    try{
3      socket = bd.createRfcommSocketToServiceRecord(
4        java.util.UUID.fromString("00001101-0000-1000-8000-00805F9B34FB")
5      );
6      socket.connect();
7      Toast.makeText(Main.this, "Robot Connected!", Toast.LENGTH_LONG).show();
8    }catch (Exception e){
9      Log.e(TAG,"Failed in connectToRobot() [" + e.getMessage() + "]");
10   }
11 }
```

και η λειτουργία της είναι η δημιουργία, στις γραμμές 3 με 5, ενός socket (υποδοχής) για να επιτευχθεί η επικοινωνία με την απομακρυσμένη συσκευή (πρωτόκολλο επικοινωνίας RFCOMM - Radio frequency communication [43]) και η αίτηση σύνδεσης, γραμμή 6, της συσκευής. Ανάλογα μπορεί να γραφτεί και η μέθοδος διακοπής της επικοινωνίας.

Για να διαπιστωθεί η επιτυχής σύνδεση της συσκευής πρέπει πρώτα να καταχωρηθούν τα σχετικά γεγονότα (events), έτσι ώστε όταν προκύψουν να είναι παρατηρήσιμα. Η δήλωση των γεγονότων αυτών τελικά τοποθετήθηκε στη μέθοδο onStart() του activity μαζί με τη κλήση της μεθόδου findRobot

[1] Όταν συνδέεται για πρώτη φορά μια συσκευή σε μια άλλη, μέσω bluetooth, γίνεται ανταλλαγή πληροφοριών διεύθυνσης και μερικές φορές και ανταλλαγή κωδικών πρόσβασης (Για το NXT brick συνήθως είναι το 0000 ή 1234). Αυτή η διαδικασία είναι γνωστή ως ζεύξη (pairing) των δύο συσκευών.

που θα ξεκινήσει τη διαδικασία μόλις ανοίξει το παράθυρο που περιέχει τις μεθόδους αυτές:

```
// set filter for receiver, for detect when connected
// and disconnected with robot via bluetooth
registerReceiver(mReceiver,
  new IntentFilter("android.bluetooth.device.action.ACL_CONNECTED")
 );
registerReceiver(mReceiver,
  new IntentFilter("android.bluetooth.device.action.ACL_DISCONNECTED")
 );
findRobot();
```

Τέλος είναι απαραίτητη η δημιουργία ενός BroadcastReceiver που θα διαβάζει τα μηνύματα του συστήματος σχετικά με την επιτυχή ή όχι σύνδεση της απομακρυσμένης συσκευής με bluetooth και θα καλεί την αντίστοιχη μέθοδο:

```
private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
  @Override
  public void onReceive(Context context,Intent intent) {
    if(intent.getAction().equals(
       "android.bluetooth.device.action.ACL_CONNECTED")) {
      handleConnected();
    }
    if(intent.getAction().equals(
       "android.bluetooth.device.action.ACL_DISCONNECTED")){
      handleDisconnected();
    }
  }
};
```

Πολύ σημαντικό είναι να δηλωθούν και τα δικαιώματα χρήσης του bluetooth στο αρχείο manifest, αλλιώς δεν θα είναι δυνατή η χρήση του bluetooth από την εφαρμογή:

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

4.2.3 Direct Commands για επικοινωνία με το NXT brick

Η επικοινωνία της κινητής συσκευής με το brick γίνεται με εντολές Direct commands, οι οποίες είναι σε μορφή αλληλουχίας byte. Συνεπώς οι εντολές αυτές έπρεπε να αντιστοιχηθούν σε κατάλληλες σταθερές και συναρτήσεις. Για παράδειγμα μερικές σταθερές είναι οι παρακάτω:

```
// Command types constants. Indicates type
// of packet being sent or received.
public static byte DIRECT_COMMAND_REPLY = (byte) 0x00;
public static byte DIRECT_COMMAND_NOREPLY = (byte) 0x80;
public static byte REPLY_COMMAND = (byte) 0x02;
```

αυτές μαζί με τις υπόλοιπες σταθερές βρίσκονται στο τελικό αρχείο NXTProtocol.java της εφαρμογής (Παράρτημα Β').

Για να αποσταλεί η εντολή χρειάζεται να σχηματιστεί η κατάλληλη αλληλουχία και να αποσταλεί. Για τις αλληλουχίες των bytes καταοκευάστηκαν συναρτήσεις, όπως η εξής:

```
1  public void setOutputState(int port, byte power, int mode,
2      int regulationMode, int turnRatio, int runState, int tachoLimit) {
3    byte[] request = {
4      DIRECT_COMMAND_NOREPLY,
5      SET_OUTPUT_STATE,
6      (byte) port,
7      power,
8      (byte) mode,
9      (byte) regulationMode,
10     (byte) turnRatio,
11     (byte) runState,
12     (byte) tachoLimit,
13     (byte) (tachoLimit >>> 8),
14     (byte) (tachoLimit >>> 16),
15     (byte) (tachoLimit >>> 24)
16   };
17   sendData(request);
18 }
```

η οποία στην ουσία φτιάχνει μια εντολή σε μορφή Direct command για την περιστροφή κινητήρα και καλεί την συνάρτηση για αποστολή, της εντολής που σύνθεσε, στο brick. Στη γραμμή 4 φαίνεται πως, είναι μια εντολή που δεν θέλει απάντηση από το brick. Στη γραμμή 5 είναι ο κωδικός εντολής (στο συγκεκριμένο για αποστολή γωνιών περιστροφής στους κινητήρες), και από τη γραμμή 6 και μετά είναι τα στοιχεία περιστροφής, όπως ποιος κινητήρας, με πόση δύναμη και για πόση ώρα θα περιστραφεί. Ανάλογα μπορούν να κατασκευαστούν και οι υπόλοιπες εντολές. Οι εντολές αυτές απαιτούν κι συναρτήσεις αποστολής τους στη συσκευή, κι έτσι έχουμε τη συνάρτηση:

```
1  public byte[] sendData(byte[] request) {
2    int lsb = request.length;
3    int msb = request.length >>> 8;
4    byte verify[] = null;
5
6    try {
7      Thread.sleep(30);
8      os.write((byte) lsb);
9      os.write((byte) msb);
10     os.write(request);
11   } catch (IOException e) {
12     Log.e("sendData", "Write failed.", e);
13     throw new RuntimeException(e);
14   } catch (InterruptedException e) {}
15
16   if(request[0] == DIRECT_COMMAND_REPLY){
17     byte[] reply = readData();
18     verify = reply;
19   }
20
21   return verify;
22 }
```

όπου, αφού υπολογιστούν τα δύο byte μήκους αλληλουχίας (γραμμές 2, 3), γίνεται η αποστολή όλης της εντολής/αλληλουχίας στο brick, με τη βοήθεια του μηχανισμού εξαιρέσεων try/catch και του OutputStream, μια κλάση της Java υπεύθυνη για την αποστολή ροής δεδομένων σε μορφή αλληλουχίας με bytes, γραμμές 8-10, σε κάποιον αποδέκτη (στην περίπτωσή μας είναι το brick). Επιπλέον στις γραμμές 16-19, αν η εντολή περιμένει απάντηση στην εντολή που αποστάλθηκε, καλεί τη συνάρτηση readData() που είναι υλοποι-

ημένη με παρόμοια με το sendData(), όπου αντί για OutputStream χρησιμοποιείται το InputStream και διαβάζει την αλληλουχία με bytes που λαμβάνει. Οι παραπάνω συναρτήσεις βρίσκονται στο τελικό αρχείο NXTCommands.java της εφαρμογής (Παράρτημα Β'). Έπειτα το αρχείο Motor.java έχει υλοποιημένες κάποιες συναρτήσεις που απλοποιούν τη διαδικασία χρησιμοποιώντας συναρτήσεις από το προηγούμενο αρχείο. Έτσι παράγονται συναρτήσεις όπως setSpeed(), getSpeed(), rotateTo() κτλ., οι οποίες θα χρησιμοποιηθούν μετέπειτα στην εφαρμογή και αφορούν κυρίως τις εντολές περιστροφής των κινητήρων.

Πρέπει να σημειωθεί ότι οι συναρτήσεις που βρίσκονται στα αρχεία NXTProtocol.java, NXTCommands.java, OutputState.java, Motor.java είναι βασισμένες κυρίως στο API του LeJOS[2], παίρνοντας ως παράδειγμα την εφαρμογή MINDdroid, που υλοποιήθηκε από την ομάδα της Lego. Είναι μια Android εφαρμογή και ο σκοπός της είναι να κινεί ένα όχημα από Lego Mindstorms [44].

4.2.4 Η πρώτη λειτουργία του συστήματος

Αφού δημιουργήθηκαν όλες οι απαραίτητες συναρτήσεις για αποστολή εντολών στο brick ήταν απαραίτητη η δοκιμή και ο έλεγχος τόσο των εντολών αυτών όσο και της υλικής κατασκευής, για να διαπιστωθούν τυχόν λάθη και δυσλειτουργίες. Οι συναρτήσεις του brick που υλοποιήθηκαν αφορούν τη λειτουργία των κινητήρων, άρα η δοκιμή αφορά την περιστροφή των κινητήρων και συνεπώς και την περιστροφή της πλάκας. Για να διατηρηθεί η κατασκευή ακέραια έπρεπε να ληφθούν υπόψη και οι περιορισμοί στις γωνίες περιστροφής της επίπεδης επιφάνειας. Για τις δοκιμές χρησιμοποιήθηκε ο αισθητήρας προσανατολισμού (γυροσκόπιο) του κινητού. Ένας αισθητήρας που αντιλαμβάνεται την περιστροφή ως προς τους τρεις άξονες της κινητής συσκευής (Σχήμα 4.7). Στη δοκιμή, όπως είναι λογικό, οι άξονες που χρησιμοποιήθηκαν είναι x και y. Μετά από τις δοκιμές προέκυψαν δύο ενδιαφέρουσες λειτουργίες.

Επακόλουθη λειτουργία Ι: Χειροκίνητη λειτουργία του ρομπότ

Για να είναι νοητά εύκολο να διαπιστωθεί ποιος κινητήρας περιστρέφεται, η επίπεδη επιφάνεια ταυτίστηκε με την οθόνη της κινητής συσκευής και ο κάθε κινητήρας με τον αντίστοιχο άξονα. Αποτέλεσμα η επίπεδη επιφάνεια να ακολουθεί τον προσανατολισμό του κινητού.

[2]LeJOS είναι μια από τις γλώσσες προγραμματισμού του NXT brick με τον κώδικα να τρέχει πάνω στο ίδιο ή σε έναν απλό ηλεκτρονικό υπολογιστή για να επικοινωνεί με αυτό.

Σχήμα 4.7: *Οι άξονες περιστροφής που διαβάζει ο αισθητήρας περιστροφής [45]*

Αφού δηλωθεί και ξεκινήσει τη λειτουργία του ο αισθητήρας, γραμμές 4-9:

```
1  private void handleConnected() {
2    try {
3      // Start orientation sensor
4      if (sManager != null) {
5        sManager.registerListener(
6          Main.this,
7          sManager.getDefaultSensor(Sensor.TYPE_ORIENTATION),
8          SensorManager.SENSOR_DELAY_UI);
9      }
10     thread = new runControl(socket.getInputStream(),socket.getOutputStream());
11   } catch (Exception e) {
12     Log.e(TAG,"Failed in handleConnected() [" + e.getMessage() + "]");
13     disconnectFromRobot();
14   }
15 }
```

αρχίζει να διαβάζει τιμές του προσανατολισμού με τη μέθοδο onSensorChanged όπου καταχωρούνται οι τιμές αυτές σε μεταβλητές που θα μπορούν να προσπελαστούν στον υπόλοιπο κώδικα. Έπειτα δημιουργείται και τρέχει ένα thread (γραμμή 10), το οποίο θα διαβάζει τις μεταβλητές προσανατολισμού, θα δημιουργήσει ένα αντικείμενο του NXT brick με τις μεθόδους που γράφτηκαν προηγουμένως και θα στέλνει τις αντίστοιχες εντολές στο brick. Οι τιμές των μεταβλητών που διαβάζονται από τον αισθητήρα είναι σε μοίρες περιστροφής του κινητού, οι τιμές αυτές μειώνονται με μια διαίρεση για να πετύχουν την αντίστοιχη περιστροφή της πλάκας, θέτονται τα όρια περιστροφής, η ταχύτητα των κινητήρων με τη μέθοδο που υλοποιήθηκε setSpeed, και τέλος με την μέθοδο rotateTo θέτονται οι γωνίες περιστροφής στον κάθε κινητήρα. Αυτή η διαδικασία επαναλαμβάνεται συνέχεια με μία καθυστέρηση μερικών milliseconds. Αυτό ρυθμίζεται από την συνάρτηση updateTime η οποία υπολογίζει το χρόνο που πέρασε από τον προηγούμενο κύκλο και θέτει την τιμή σε μια μεταβλητή για να συγκριθεί με τον απαιτούμενο χρόνο που τίθεται.

Έτσι τελικά προέκυψε μια χειροκίνητη λειτουργία του συστήματος, δηλα-

δή βλέπουμε τη σφαίρα και προσπαθούμε να την ισορροπήσουμε στο κέντρο της πλάκας, μέσω της περιστροφής τους κινητού, έτσι όπως θα κάναμε αν είχαμε την πλάκα στα χέρια μας. Πρέπει να σημειωθεί πως λόγω της κατασκευής οι άξονες περιστροφής των κινητήρων είναι οι δύο διαγώνιοι της επίπεδης επιφάνειας.

Επακόλουθη λειτουργία ΙΙ : Ισοστάθμιση

Μετά από τον προηγούμενο πειραματισμό προέκυψε η σκέψη να μπει το κινητό πάνω στην επίπεδη επιφάνεια για να παίξει τον ρόλο του γνωστού αλφαδιού δύο διαστάσεων, δηλαδή όταν μπει το κινητό στην επιφάνεια, σε αυτήν την λειτουργία, θα ισορροπήσει σε απόλυτα οριζόντια θέση την επιφάνεια του συστήματος. Όπως προκύπτει, στις μεταβολές της κλίσης όλου του συστήματος, η επίπεδη επιφάνεια θα είναι πάντα σε οριζόντια θέση. Η λειτουργία αυτή θυμίζει τον δίσκο του σερβιτόρου και την προσπάθεια του να τον κρατήσει οριζόντια παρά τις μεταβολές στο σώμα του.

Μια μικρή σημείωση: σε αντίθεση με την προηγούμενη λειτουργία, όπου η κλίση της επίπεδης επιφάνειας ακολουθούσε την κλίση της κινητής συσκευής, σε αυτήν τη λειτουργία η περιστροφή την επίπεδης επιφάνειας είναι αντίθετη στις περιστροφές του κινητού.

4.2.5 Αλγόριθμοι OpenCV για τον εντοπισμό της σφαίρας

Αφού διορθώθηκαν κάποια λάθη στις συναρτήσεις και φάνηκε να λειτούργει σωστά το σύστημα ως προς τις συναρτήσεις περιστροφής κινητήρων, προχωρήσαμε στο σημαντικό κομμάτι, αυτό της αναγνώρισης της σφαίρας πάνω στην επίπεδη επιφάνεια. Για την αναγνώριση της σφαίρας χρησιμοποιήθηκε η βιβλιοθήκη OpenCV4Android.

Το περιεχόμενο των δύο αρχείων, AutoControlView.java και AutoControlViewBase.java, δίνεται από την OpenCV. Τα αρχεία περιέχουν την ενεργοποίηση των μεθόδων και μέσα σε αυτά τα αρχεία προστέθηκε ο κώδικας για την αναγνώριση της σφαίρας και ο μετέπειτα κώδικας.

Το αρχείο AutoControlViewBase.java περιλαμβάνει την κλάση που είναι υπεύθυνη για την ενεργοποίηση της κάμερας, της λήψης εικόνας και της εμφάνισης της εικόνας, μετά την επεξεργασία, στην οθόνη. Στο αρχείο έγιναν μικρές αλλαγές.

Το αρχείο AutoControlView.java περιλαμβάνει την κλάση που περιέχει το κυρίως κώδικα και είναι η επέκταση της προηγούμενης κλάσης. Ο κώδικας που γράφτηκε για τον εντοπισμό της σφαίρας είναι ο εξής:

```
1 //retrieve capture data with RGB format
2 capture.retrieve(matRGB, Highgui.CV_CAP_ANDROID_COLOR_FRAME_BGR);
3
```

```
4  //Cut video width => square capture frame
5  int cutWidth = (matRGB.width() - matRGB.height())/2;
6  matRGBsq = matRGB.submat(
7    0, matRGB.height(), cutWidth, matRGB.width()-cutWidth);
8
9  //tranform video capture to to HSV format
10 Imgproc.cvtColor(matRGBsq, matHSV, Imgproc.COLOR_BGR2HSV);
11
12 //Koboume otidhpote den einai kokkino
13 Core.inRange(matHSV,new Scalar(0, 100, 30),
14   new Scalar(15, 255, 255), thresholded);
15 Core.inRange(matHSV, new Scalar(150, 100, 30),
16   new Scalar(180, 255, 255), thresholded2);
17 Core.bitwise_or(thresholded, thresholded2, matFinal);
18
19 //filtraroume gia kalutera apotelesmata twn kuklwn
20 Imgproc.GaussianBlur(matFinal, matFinal, new Size(3, 3), 0);
21
22 // Variables for HoughCircles search
23 Mat circles = new Mat();
24 double minDist = matFinal.height();
25 int cannyUpperThreshold = 200;
26 int accumulatorThreshold = 20;
27 int minRadius = 15;
28 int maxRadius = 30;
29
30 Imgproc.HoughCircles(matFinal, circles,Imgproc.CV_HOUGH_GRADIENT, 1.5,
31   minDist, cannyUpperThreshold, accumulatorThreshold, minRadius,
32   maxRadius);
33
34 // if ball founded
35 if (circles.cols() > 0){
36   // retrieve ball coords from camera
37   double vCircle[] = circles.get(0,0);
38   double i = vCircle[0];
39   double j = vCircle[1];
40
41   // draw circle for founded ball
42   Point ptBall = new Point(Math.round(i), Math.round(j));
43   Core.circle(matFinal, ptBall, 25, new Scalar(255,255,255), 3);
44 }
45
46 // create bitmap image for output
47 Bitmap bmp = Bitmap.createBitmap(
48   matFinal.width(), matFinal.height(), Bitmap.Config.ARGB_8888);
49
50 try {
51   // transform mat to bitmap and return to display
52   Utils.matToBitmap(matFinal, bmp);
53   return bmp;
54 } catch(Exception e) {
55   Log.e("AutoControlView", "fail in processFrame() " + e.getMessage());
56   bmp.recycle();
57   return null;
58 }
```

όπου στη γραμμή 2 η συνάρτηση τραβάει το στιγμιότυπο από την κάμερα και το αποθηκεύει. Στις γραμμές 5 με 7 μεταοχηματίζεται το στιγμιότυπο από ορθογώνιο σε τετράγωνο. Στη γραμμή 10 μετατρέπεται το στιγμιότυπο από το χρωματικό μοντέλο BGR σε HSV, για διευκόλυνση εντοπισμού της οφαίρας. Στις γραμμές 13 με 17 γίνεται η εικόνα δίχρωμη με άοπρο χρώμα να είναι το κόκκινο χρώμα της οφαίρας και μαύρο όλα τα υπόλοιπα χρώματα. Στη γραμμή 20 εφαρμόζεται το φίλτρο εξομάλυνοης Gaussian, το οποίο βελτιώνει τα αποτελέσματα του μεταοχηματισμού Hough, γραμμές 23-32. Ο μεταοχη-

ματισμός Hough είναι μια τεχνική εντοπισμού διαφόρων σχημάτων, και στην περίπτωσή μας των κύκλων. Αφού εντοπιστεί η σφαίρα, σχεδιάζεται ένας κύκλος πάνω στο στιγμιότυπο (γραμμές 35-44) και στη συνέχεια εκτυπώνεται στην οθόνη του κινητού (γραμμές 47-58). Η έξοδος του μετασχηματισμού Hough είναι οι συντεταγμένες (κέντρου) όλων των κυκλικών αντικειμένων που εντοπίζονται. Αφού λοιπόν ρυθμίστηκε η ευαισθησία του μετασχηματισμού, πήραμε σαν αποτέλεσμα τις συντεταγμένες μόνο του κέντρου της σφαίρας.

4.2.6 Υλοποίηση PID ελεγκτών για υπολογισμό γωνιών και αποστολή εντολών στο NXT brick

Αφού γράφτηκε ο κώδικας για τον εντοπισμό της σφαίρας και λειτούργησε ικανοποιητικά, γράφτηκε το τελευταίο κομμάτι της εφαρμογής, ο υπολογισμός των γωνιών περιστροφής. Για τον υπολογισμό των γωνιών χρησιμοποιήθηκαν οι PID ελεγκτές.

Το σύστημα συντεταγμένων του στιγμιότυπου της κάμερας δε συμπίπτει με το νοητό σύστημα συντεταγμένων της επίπεδης επιφάνειας (το σύστημα συντεταγμένων που ορίζεται από τις περιστροφές κινητήρων, δηλαδή οι διαγώνιοι της επιφάνειας και κέντρο το σημείο που διασταυρώνονται οι διαγώνιες). Για το λόγο αυτό έγινε δύο μετατροπές στις συντεταγμένες του κέντρου της σφαίρας στο στιγμιότυπο. Η πρώτη μετατροπή ήταν η μεταφορά της αρχής των αξόνων από την πάνω αριστερή γωνία του στιγμιότυπου, στο κέντρο του:

$$x' = x - frame_{width}/2 \tag{4.1}$$

$$y' = -(y - frame_{height}/2) \tag{4.2}$$

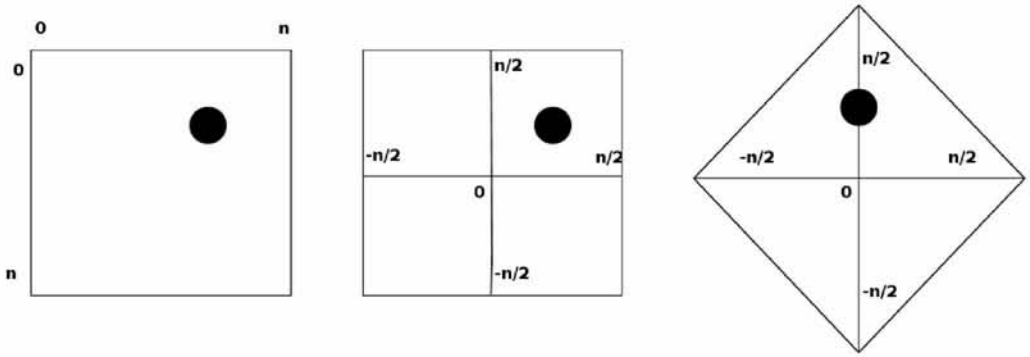
Η δεύτερη μετατροπή ήταν οι συντεταγμένες να περιστραφούν κατά 45 μοίρες αντίθετα με τη φορά του ρολογιού (Σχήμα 4.8). Έτσι γράφτηκαν δυο συναρτήσεις, οι rotetedCoordX και rotetedCoordY που υπολογίζουν τις καινούριες συντεταγμένες της σφαίρας, σύμφωνα με τις εξής μετατροπές [46]:

$$x'' = \cos(\pi/4) * x' - \sin(\pi/4) * y' \tag{4.3}$$

$$y'' = \sin(\pi/4) * x' + \cos(\pi/4) * y' \tag{4.4}$$

Αφού έγιναν οι μετατροπές στις συντεταγμένες στη συνέχεια υπολογίστηκαν οι γωνίες περιστροφής των κινητήρων:

```
1 // current error
2 ErrorCurX = desiredX - newX;
3 // integrate error
4 ErrorIntX = ErrorIntX + ErrorCurX * timeInterval;
5 // derivative error
6 double ErrorDerX = (ErrorCurX - ErrorPrevX)/timeInterval;
```



Σχήμα 4.8: *Μετατροπή των συντεταγμένων της σφαίρας, περιστροφή κατά 45 μοίρες*

```
7 ErrorPrevX = ErrorCurX;
8 // calculate angle for rotate motor
9 int angleC = (int) Math.round((Kp*ErrorCurX) + (Ki*ErrorIntX) + (Kd*ErrorDerX
      ));
```

όπου στις γραμμές 2 με 7 υπολογίζονται οι τρεις όροι του ελεγκτή, οι οποίοι πολλαπλασιάζονται στη συνέχεια με την αντίστοιχη σταθερά κέρδους, γραμμή 9. Το αποτέλεσμα, δηλαδή η γωνία περιστροφής αποστέλλεται στο brick και ξεκινάει η διαδικασία πάλι από την αρχή με το καινούριο στιγμιότυπο.

Μετά από αρκετούς πειραματισμούς με διάφορες σταθερές κέρδους του ελεγκτή και διάφορες υφές της επίπεδης επιφάνειας, το σύστημα κατάφερε να ισορροπήσει τη σφαίρα στο κέντρο της επιφάνειας. Ο κώδικας της εφαρμογής βρίσκεται στο παράρτημα Β΄.

Αφού ολοκληρώθηκε ο προγραμματισμός του ρομπότ, ο κώδικας τακτοποιήθηκε σε μια ενιαία εφαρμογή που περιέχει και τις τρεις λειτουργίες, δίνοντας στο χρήστη να επιλέξει το πείραμα που θέλει να τρέξει, όπως φαίνεται στα παρακάτω σχήματα. Όλα τα screenshots της εφαρμογής βρίσκονται στο παράρτημα Γ΄.

Σχήμα 4.9: *Το γραφικό περιβάλλον του κεντρικού activity της εφαρμογής*

Σχήμα 4.10: *Το γραφικό περιβάλλον της λειτουργίας του συστήματος*

Κεφάλαιο 5

Συμπεράσματα

Η εργασία ολοκληρώθηκε και ο σκοπός του ρομπότ έχει επιτευχθεί. Το σύστημα που ισορροπεί τη σφαίρα πάνω σε μια επιφάνεια, ένα πρόβλημα ασταθές και δύσκολο στην υλοποίηση, έχει ολοκληρωθεί και ανταποκρίνεται αρκετά ικανοποιητικά, παρόλο που τα υλικά που χρησιμοποιήθηκαν ήταν αρκετά φθηνά σε σχέση με άλλα ρομποτικά συστήματα, συνεπώς και λιγότερης ακριβείας και αξιοπιστίας. Ωστόσο η διάρκεια μέχρι να ισορροπηθεί η σφαίρα στο κέντρο της επίπεδης επιφάνειας είναι αρκετά μεγάλη, κάτι το οποίο δεν είναι απόλυτα ικανοποιητικό και θα μπορούσε να βελτιωθεί με μια πιο αξιόπιστη ρομποτική κατασκευή. Πιθανές αιτίες που ευθύνονται στη μειωμένη απόδοση του συστήματος είναι κάποιες από τις παρακάτω:

- Οι καθυστερήσεις λόγω της επικοινωνίας, κινητής συσκευής και brick, μέσω bluetooth, αποτελούν σημαντικό λόγο μειωμένης απόδοσης του συστήματος. Οι καθυστερήσεις αυτές θα μπορούσαν να εξαλειφθούν με τη χρήση μιας υπολογιστικής μονάδας συνδεδεμένη απευθείας με τους κινητήρες.

- Η επεξεργαστική ισχύς της κινητής συσκευής δεν είναι αρκετή. Οι αλγόριθμοι της βιβλιοθήκης OpenCV είναι αλγόριθμοι επεξεργασίας εικόνας και βίντεο και χρειάζονται μεγάλη επεξεργαστική ισχύ και αρκετό χώρο προσωρινής μνήμης. Άρα θα μπορούσε να χρησιμοποιηθεί μια ισχυρότερη υπολογιστική μονάδα ελέγχου ή να χρησιμοποιηθούν περισσότερο αποδοτικοί αλγόριθμοι.

- Η επίπεδη επιφάνεια είναι μικρή. Όσο μικρότερη είναι η επιφάνεια τόσο λιγότερη ανοχή υπάρχει στα λάθη, γιατί λιγοστεύει ο χώρος κίνησης της σφαίρας, άρα και η απόκριση του συστήματος πρέπει να γίνει γρηγορότερη. Συνεπώς μια κατασκευή με μεγαλύτερη επίπεδη επιφάνεια θα είχε ικανοποιητικότερα αποτελέσματα.

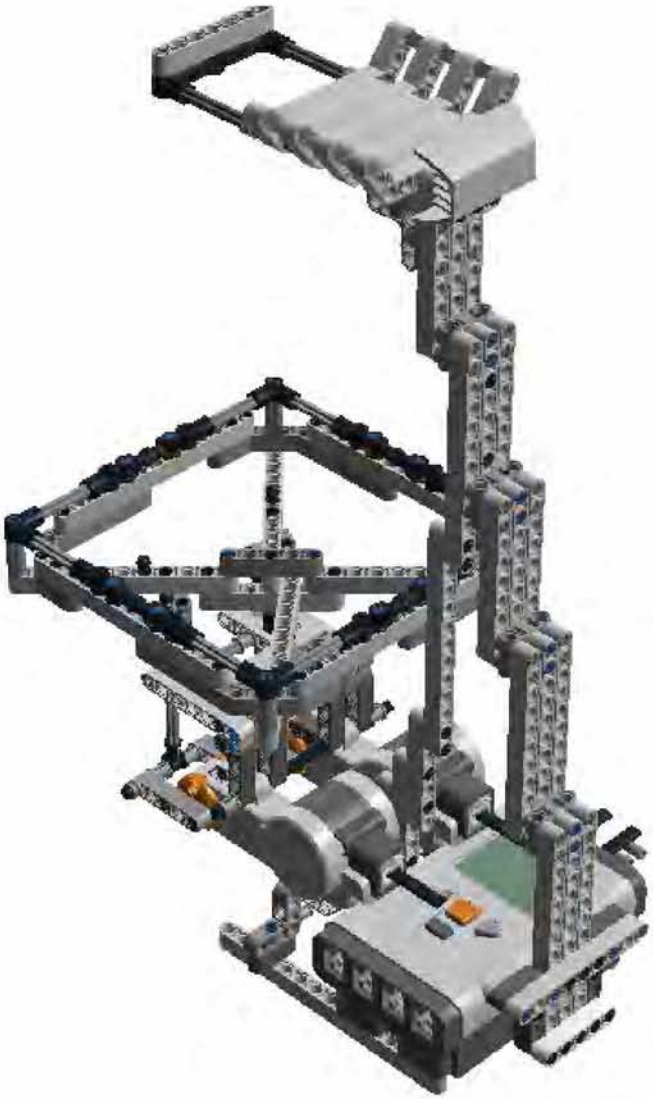
- Η κατασκευή. Γνωρίζουμε ότι οι κινητήρες της κατασκευής έχουν ακρίβεια +/- 1 μοίρα, συνεπώς το εύρος οφάλματος περιστροφής 2 μοιρών μπορεί να στοιχίσει τη σωστή λειτουργία σε ένα ασταθές σύστημα, όπως αυτό που υλοποιήθηκε. Άρα για καλύτερα αποτελέσματα η κατασκευή θα μπορούσε να ακολουθήσει διαφορετική αρχιτεκτονική όπου η γωνία περιστροφής του κάθε κινητήρα να είναι αναλογικά μεγαλύτερη από τη γωνία περιστροφής της επίπεδης επιφάνειας και έτσι να ελαττωθεί σημαντικά το σφάλμα των κινητήρων.

Μελλοντική επέκταση της εργασίας θα μπορούσε να είναι η υλοποίηση σου συστήματος σε ρομποτικά συστήματα μεγαλύτερης ακρίβειας και αξιοπιστίας. Επίσης, μια ακόμα επέκταση θα μπορούσε να γίνει, με το να παρέχεται στο χρήστη η δυνατότητα να διαλέγει το επιθυμητό σημείο ισορροπίας ή ακόμα και να ορίζει τροχιές μετακίνησης της σφαίρας πάνω στην επίπεδη επιφάνεια.

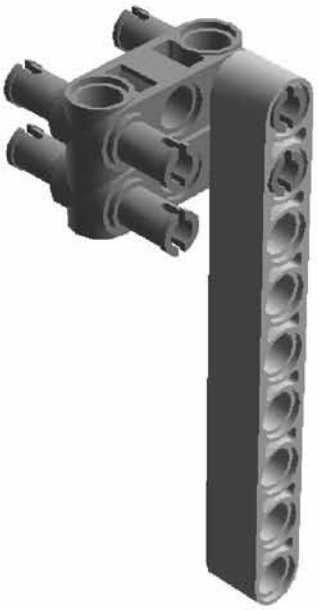
Η τεχνολογίας των συστημάτων αυτόματου ελέγχου ισορροπίας μπορεί να εφαρμοστεί σε πολλές ρομποτικές κατασκευές όπου απαιτείται η ισορροπία. Χαρακτηριστικό παράδειγμα είναι η δυνατότητα διατήρησης αντικειμένων σε ισορροπία πάνω σε ένα δίσκο που μετακινεί κάποιο ρομπότ, ειδικά όταν αυτό μετακινείται σε ανώμαλο έδαφος ή με μη σταθερό τρόπο.

Παράρτημα Α΄

Οδηγίες κατασκευής του ρομπότ



Σχήμα Α΄.1: *Βήμα 1*



Σχήμα Α΄.2: *Βήμα 2*

Σχήμα Α΄.3: *Βήμα 3*

Σχήμα Α΄.4: *Βήμα 4*

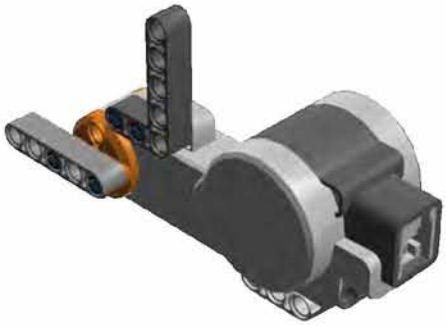
Σχήμα Α΄.5: *Βήμα 5*

Σχήμα Α΄.6: *Βήμα 6*

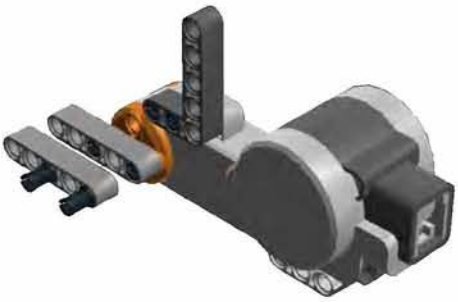
Σχήμα Α΄.7: *Βήμα 7*

Σχήμα Α΄.8: *Βήμα 8*

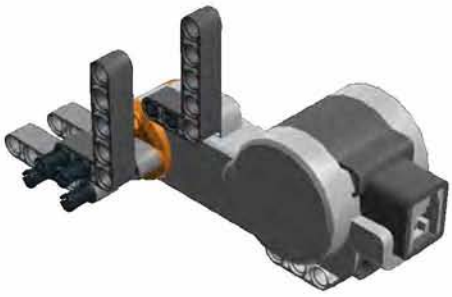
Σχήμα Α΄.9: *Βήμα 9*



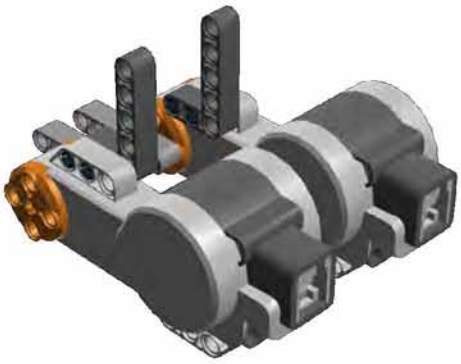
Σχήμα Α΄.10: *Βήμα 10*



Σχήμα Α΄.11: *Βήμα 11*



Σχήμα Α΄.12: *Βήμα 12*



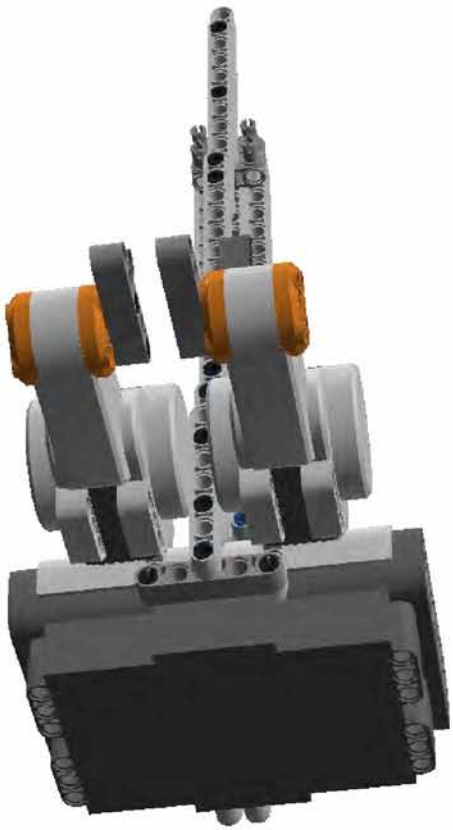
Σχήμα Α΄.13: *Βήμα 13*

Σχήμα Α΄.14: *Βήμα 14*

Σχήμα Α΄.15: *Βήμα 15*

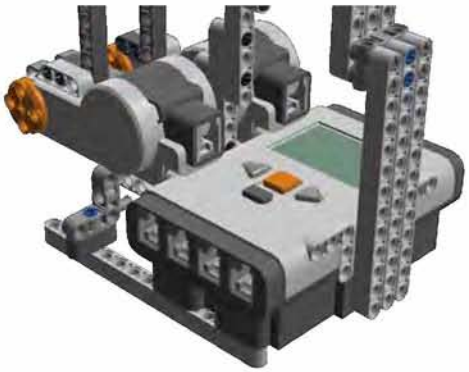
Σχήμα Α΄.16: *Βήμα 16*

Σχήμα Α΄.17: *Βήμα 17*

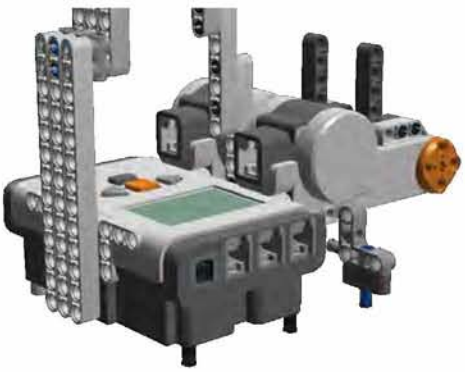


Σχήμα Α΄.18: *Βήμα 18*

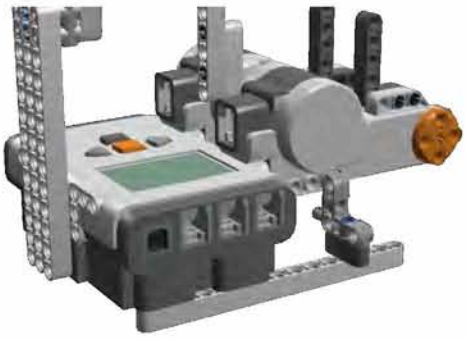
Σχήμα Α΄.19: *Βήμα 19*



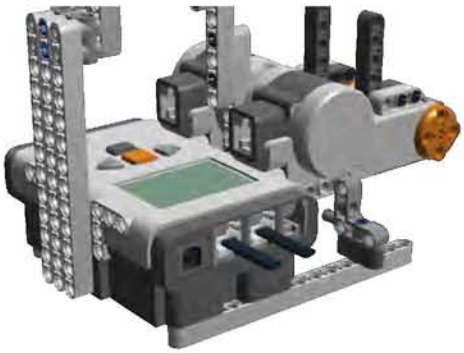
Σχήμα Α΄.20: *Βήμα 20*



Σχήμα Α΄.21: *Βήμα 21*

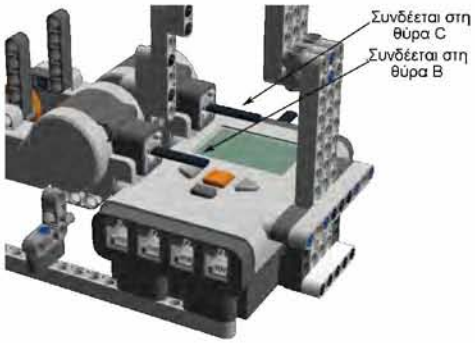


Σχήμα Α΄.22: *Βήμα 22*

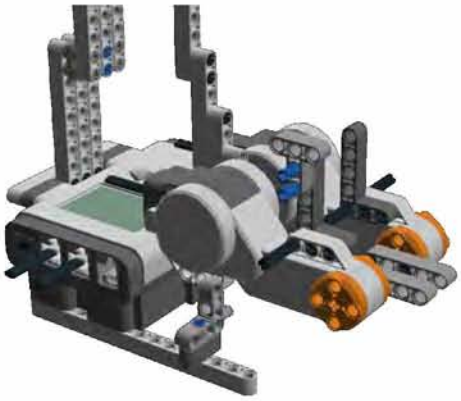


Σχήμα Α΄.23: *Βήμα 23*

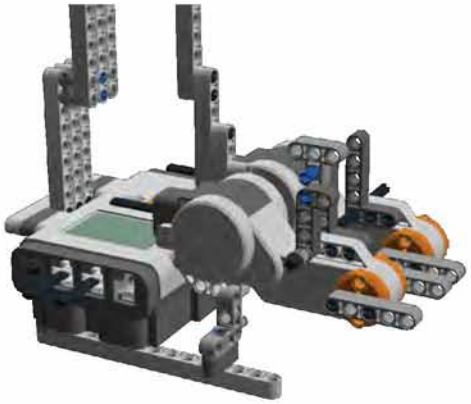
Σχήμα Α΄.24: *Βήμα 24*



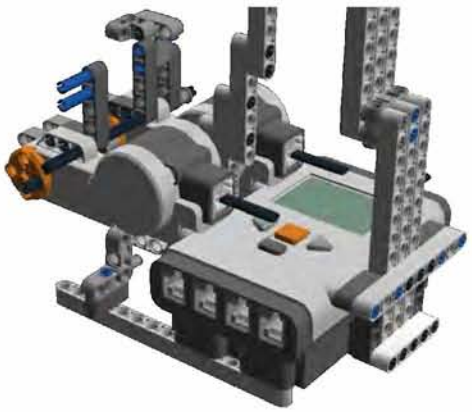
Σχήμα Α΄.25: *Βήμα 25*



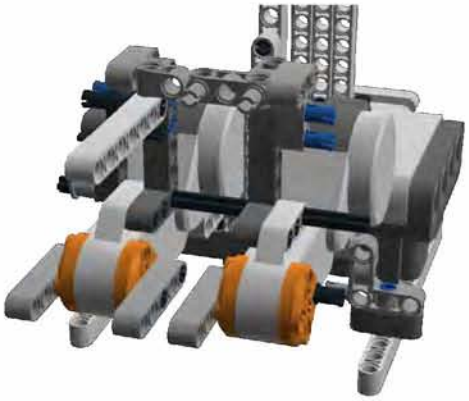
Σχήμα Α΄.26: *Βήμα 26*



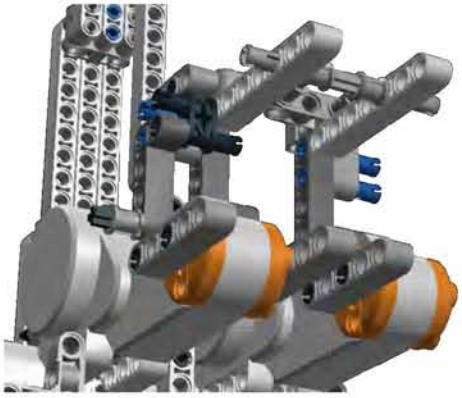
Σχήμα Α΄.27: *Βήμα 27*



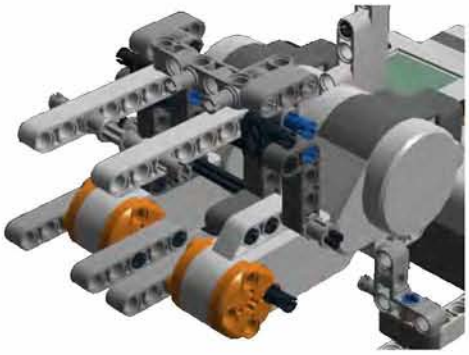
Σχήμα Α΄.28: *Βήμα 28*



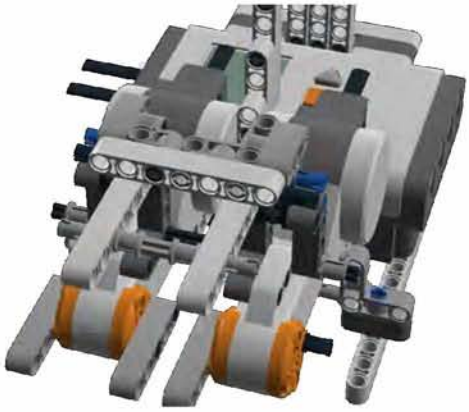
Σχήμα Α΄.29: *Βήμα 29*



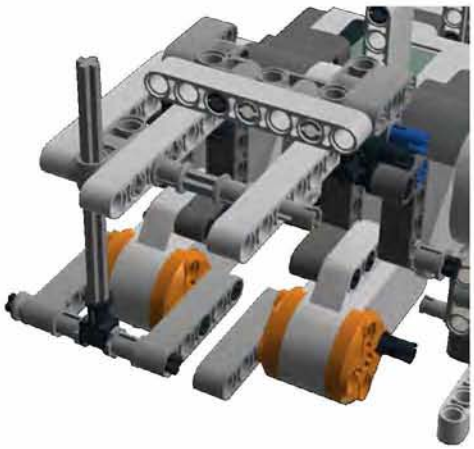
Σχήμα Α΄.30: *Βήμα 30*



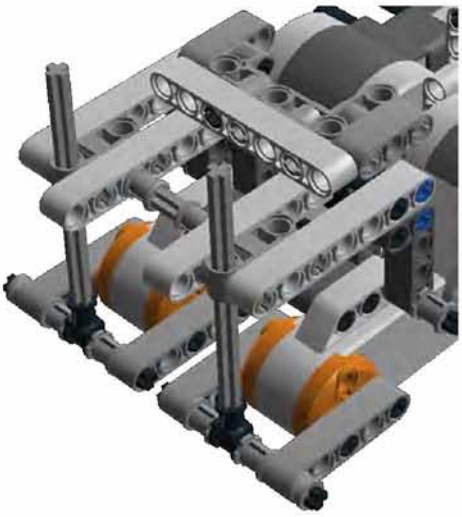
Σχήμα Α΄.31: *Βήμα 31*



Σχήμα Α΄.32: *Βήμα 32*



Σχήμα Α΄.33: *Βήμα 33*



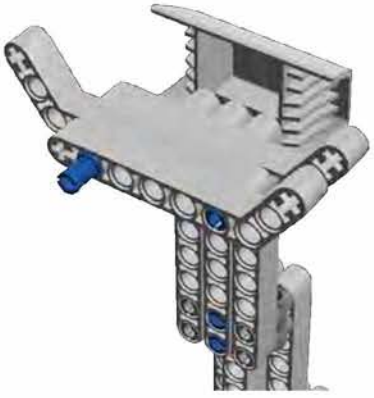
Σχήμα Α΄.34: *Βήμα 34*

Σχήμα Α΄.35: *Βήμα 35*

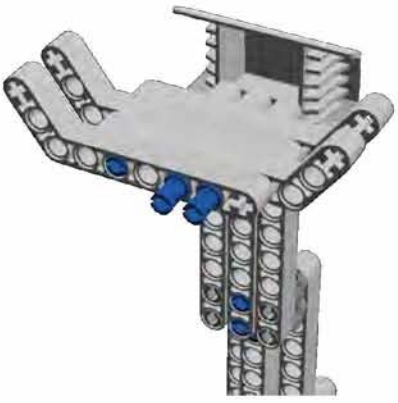
Σχήμα Α΄.36: *Βήμα 36*

Σχήμα Α΄.37: *Βήμα 37*

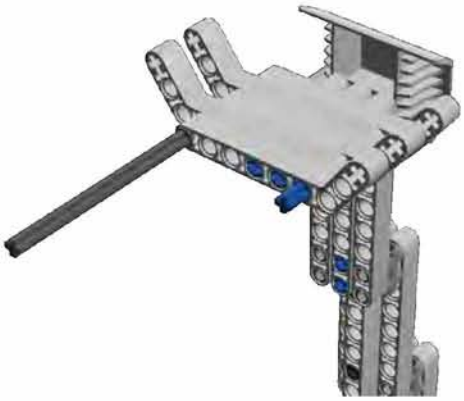
Σχήμα Α΄.38: *Βήμα 38*



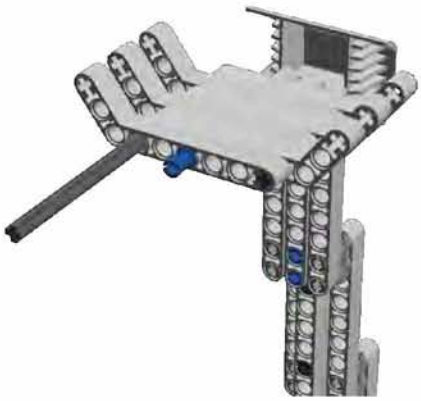
Σχήμα Α΄.39: *Βήμα 39*



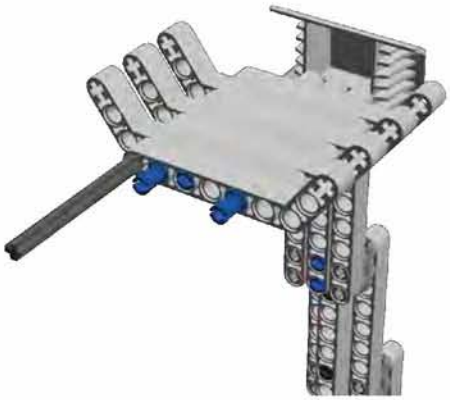
Σχήμα Α΄.40: *Βήμα 40*



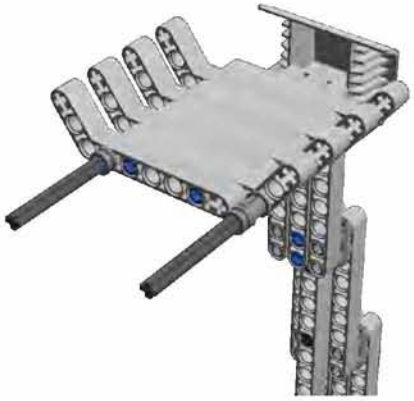
Σχήμα Α΄.41: *Βήμα 41*



Σχήμα Α΄.42: *Βήμα 42*

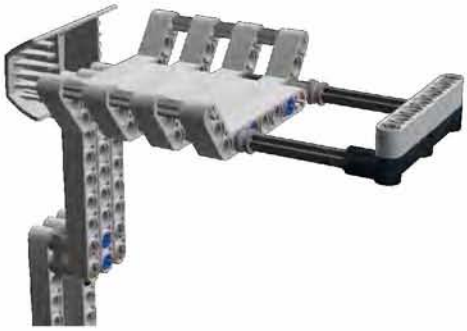


Σχήμα Α΄.43: *Βήμα 43*

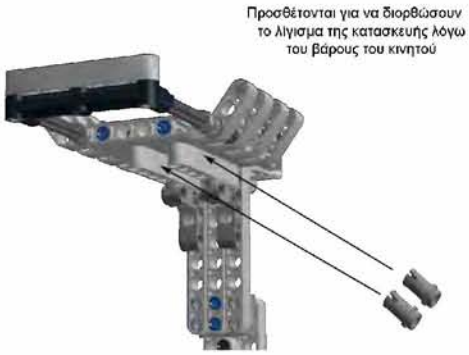


Σχήμα Α΄.44: *Βήμα 44*

Σχήμα Α΄.45: *Βήμα 45*



Σχήμα Α΄.46: *Βήμα 46*



Σχήμα Α΄.47: *Βήμα 47*

Σχήμα Α΄.48: *Βήμα 48*

Σχήμα Α΄.49: *Βήμα 49*

Σχήμα Α΄.50: *Βήμα 50*

Σχήμα Α΄.51: *Βήμα 51*

Σχήμα Α΄.52: *Βήμα 52*

Σχήμα Α΄.53: *Βήμα 53*

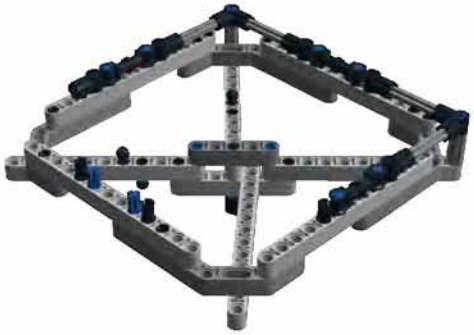
Σχήμα Α΄.54: *Βήμα 54*

Σχήμα Α΄.55: *Βήμα 55*

Σχήμα Α΄.56: *Βήμα 56*

Σχήμα Α΄.57: *Βήμα 57*

Σχήμα Α΄.58: Βήμα 58



Σχήμα Α΄.59: Βήμα 59

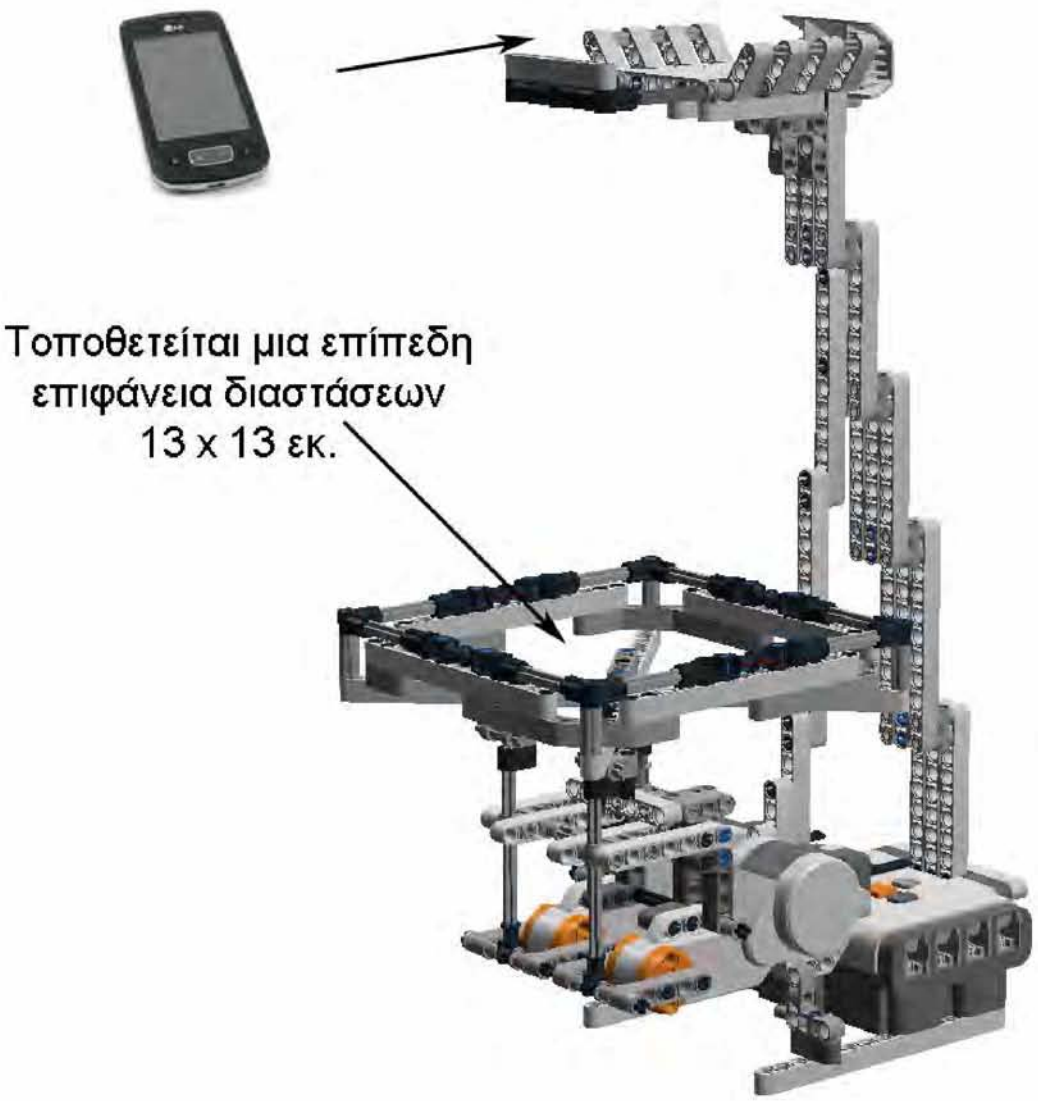
Σχήμα Α΄.60: Βήμα 60

Σχήμα Α΄.61: Βήμα 61

Σχήμα Α΄.62: Βήμα 62

Σχήμα Α΄.63: Βήμα 63

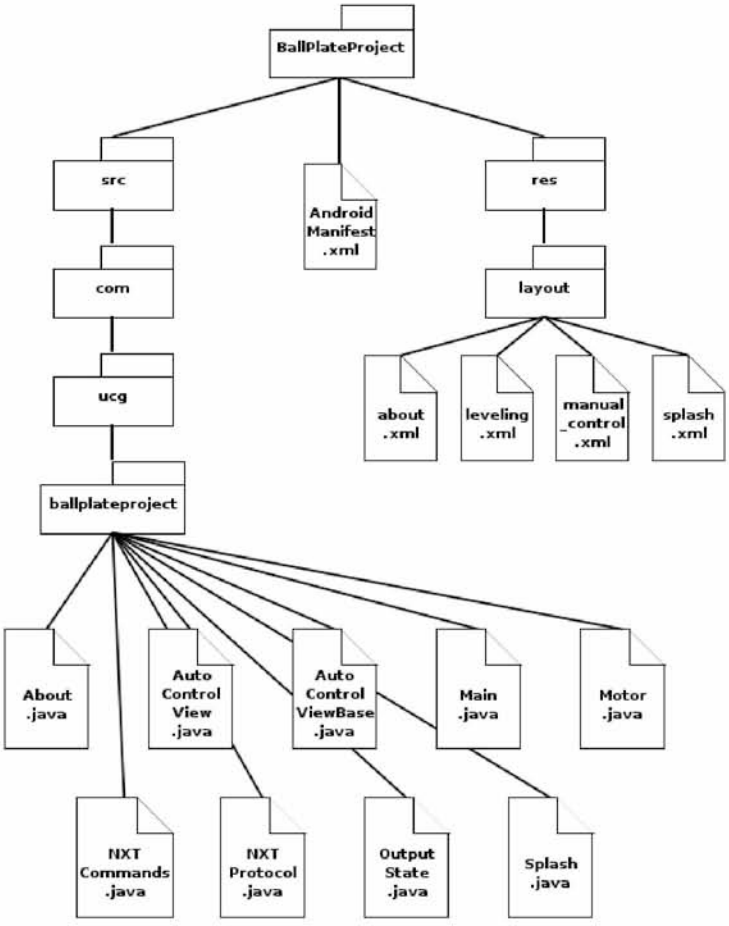
Σχήμα Α΄.64: *Βήμα 64*



Σχήμα Α΄.65: *Τελικό βήμα*

Παράρτημα Β΄

Πηγαίος κώδικας της εφαρμογής



Σχήμα Β΄.1: *Δομή του πηγαίου κώδικα*

AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.ucg.ballplateproject"
  android:versionCode="1"
  android:versionName="1.0" >

  <uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="15" />

  <application
    android:icon="@drawable/ucg_logo"
    android:label="Ball Plate Project">

    <activity
      android:name=".Splash"
      android:label="Ball Plate Project"
      android:screenOrientation="portrait">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>

    <activity
      android:name=".Main"
      android:screenOrientation="portrait" >
    </activity>

    <activity
      android:name=".About"
      android:screenOrientation="portrait"
      android:label="About"
      android:theme="@android:style/Theme.Dialog">
    </activity>
  </application>

  <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
  <uses-permission android:name="android.permission.BLUETOOTH" />
  <uses-permission android:name="android.permission.CAMERA"/>
  <uses-feature android:name="android.hardware.camera" />
  <uses-feature android:name="android.hardware.camera.autofocus" />
</manifest>
```

about.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent" >

  <LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:layout_marginTop="20dp"
    android:layout_marginBottom="30dp">

    <LinearLayout
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:orientation="horizontal" >

      <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ucg_logo"
        android:layout_margin="10dp"/>
      <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_marginTop="15dp">

        <TextView
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:text="UNIVERSITY OF CENTRAL GREECE"
          android:textSize="14dp" />
        <TextView
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:text="DEPARTMENT OF COMPUTER SCIENCE AND BIOMEDICAL INFORMATICS"
          android:textSize="10dp"/>
      </LinearLayout>
    </LinearLayout>

    <LinearLayout
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:orientation="vertical"
      android:layout_margin="10dp">

      <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="This application is part of graduate thesis"
        android:layout_margin="5dp"/>
      <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Title: Construction and programming robot"
        android:layout_margin="5dp"/>
      <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Creator: Xalkidoy Natalia"
        android:layout_margin="5dp"/>
      <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Supervisor: Plagianakos Vasilis"
        android:layout_margin="5dp"/>
      <TextView
```

```
69          android:layout_width="wrap_content"
70          android:layout_height="wrap_content"
71          android:text="2012"
72          android:layout_gravity="center"
73          android:layout_margin="10dp"/>
74      </LinearLayout>
75    </LinearLayout>
76  </ScrollView>
```

leveling.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical" >

  <ImageView
    android:layout_width="wrap_content"
    android:layout_height="250dp"
    android:src="@drawable/arrow" />
  <TextView
    android:id="@+id/tvSensorReads"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text=""
    android:layout_gravity="center"
    android:textSize="20dp"/>
</LinearLayout>
```

manual_control.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical" >

  <ImageView
    android:layout_width="wrap_content"
    android:layout_height="250dp"
    android:src="@drawable/arrow" />
  <TextView
    android:id="@+id/tvSpeed"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Select max motors speed (0 degree/sec)"
    android:layout_marginLeft="10dp"/>
  <SeekBar
    android:id="@+id/sbSpeed"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp" />
  <TextView
    android:id="@+id/tvAngle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Select max motors angle (0 degree)"
    android:layout_marginLeft="10dp"/>
  <SeekBar
    android:id="@+id/sbAngle"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp" />
  <TextView
    android:id="@+id/tvSensorReadings"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text=""
    android:layout_gravity="center"
    android:textSize="20dp"/>
</LinearLayout>
```

splash.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical">

  <LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >

    <ImageView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:src="@drawable/ucg_logo"
      android:layout_margin="10dp"/>
    <LinearLayout
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:orientation="vertical"
      android:layout_marginTop="15dp">

      <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="UNIVERSITY OF CENTRAL GREECE"
        android:textSize="14dp" />
      <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="DEPARTMENT OF COMPUTER SCIENCE AND BIOMEDICAL INFORMATICS"
        android:textSize="10dp"/>
    </LinearLayout>
  </LinearLayout>

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:layout_marginBottom="20dp"
    android:text="Ball Plate Project"
    android:textSize="40dp"
    android:layout_gravity="center" />
  <LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:layout_margin="20dp">

    <Button
      android:id="@+id/bLeveling"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_margin="5dp"
      android:text="Leveling" />
    <Button
      android:id="@+id/bAutoControl"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_margin="5dp"
      android:text="Auto Control\nCAMERA" />
    <Button
      android:id="@+id/bManualControl"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_margin="5dp"
      android:text="Manual Control\nACCELEROMETER" />
```

```
69      <Button
70        android:id="@+id/bAbout"
71        android:layout_width="fill_parent"
72        android:layout_height="wrap_content"
73        android:layout_margin="5dp"
74        android:text="About" />
75    </LinearLayout>
76  </LinearLayout>
```

About.java

```
1  package com.ucg.ballplateproject;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5
6  public class About extends Activity {
7    @Override
8    public void onCreate(Bundle savedInstanceState) {
9      super.onCreate(savedInstanceState);
10     setContentView(R.layout.about);
11   }
12 }
```

AutoControlView.java

```
1  package com.ucg.ballplateproject;
2
3  import java.io.InputStream;
4  import java.io.OutputStream;
5
6  import org.opencv.android.Utils;
7  import org.opencv.core.Core;
8  import org.opencv.core.Mat;
9  import org.opencv.core.Point;
10 import org.opencv.core.Scalar;
11 import org.opencv.core.Size;
12 import org.opencv.highgui.Highgui;
13 import org.opencv.highgui.VideoCapture;
14 import org.opencv.imgproc.Imgproc;
15
16 import android.content.Context;
17 import android.graphics.Bitmap;
18 import android.util.Log;
19 import android.view.SurfaceHolder;
20
21 public class AutoControlView extends AutoControlViewBase {
22   // Mat variables for save video frames (input and edited frames)
23   private Mat matRGB;
24   private Mat matFinal;
25   private Mat matRGBsq;
26   private Mat matHSV;
27   private Mat thresholded;
28   private Mat thresholded2;
29   // end Mat variables
30
31   // Desired x y positions
32   double desiredX = 0;
33   double desiredY = 0;
34
35   // PID controller gain values
36   double Kp = 3.5;
37     double Ki = 1.5;
38     double Kd = 2.5;
39
40   // Variable for current error
41   double ErrorCurX = 0;
42   double ErrorCurY = 0;
43   // Variable for previous error
44   double ErrorPrevX = 0;
45   double ErrorPrevY = 0;
46   // Variable for integrate error
47   double ErrorIntX = 0;
48   double ErrorIntY = 0;
49
50   // variable computing time interval between movement
51   long systemLastTime = System.currentTimeMillis();
52   double timeInterval = 0;
53
54   // Motors
55   public static final byte MOTOR_B = 0x01;
56   public static final byte MOTOR_C = 0x02;
57   NXTCommands nxtCommands;
58   Motor motorB = null;
59   Motor motorC = null;
60   // max angle for rotate motors
61   int maxAngle = 10;
62
63   public AutoControlView(Context context, InputStream is, OutputStream os) {
64     super(context);
65     // create object for nxt commands class and set input output streams
66     nxtCommands = new NXTCommands(is, os);
67
68     // create objects for servo motors
```

```
69      motorB = new Motor(MOTOR_B);
70      motorC = new Motor(MOTOR_C);
71      // set motors streams of nxt
72      motorB.setStreams(is, os);
73      motorC.setStreams(is, os);
74      // set motors speeds
75      motorB.setSpeed(30);
76      motorC.setSpeed(30);
77      // pairnoyme dyo fores th gwnia toy enos kinhthra gia logoys ry8mishs
78      // (to getTachoCount ana pente fores diavazei plhroforia apo to nxt poy
79      // xrhsimopoieitai sto rotate kai tis alles fores apo mia metavlhth
80      // poy thn apo8hkeyei, opote gia na mhn diavazei gwnies taytoxrona
81      // gia toys dyo kinhthres travame 2 gwnies arxika)
82      motorB.getTachoCount();
83      motorB.getTachoCount();
84    }
85
86    @Override
87    public void surfaceCreated(SurfaceHolder holder) {
88      synchronized (this) {
89        // initialize Mats before usage
90        matFinal = new Mat();
91        matRGB = new Mat();
92        matRGBsq = new Mat();
93        matHSV = new Mat();
94        thresholded = new Mat();
95        thresholded2 = new Mat();
96      }
97      super.surfaceCreated(holder);
98    }
99
100   @Override
101   protected Bitmap processFrame(VideoCapture capture) {
102     //retrieve capture data with RGB format
103     capture.retrieve(matRGB, Highgui.CV_CAP_ANDROID_COLOR_FRAME_BGR);
104
105     //Cut video width => square capture frame
106     int cutWidth = (matRGB.width() - matRGB.height())/2;
107     matRGBsq = matRGB.submat(0, matRGB.height(), cutWidth, matRGB.width()-cutWidth
          );
108
109     //tranform video capture to to HSV format
110     Imgproc.cvtColor(matRGBsq, matHSV, Imgproc.COLOR_BGR2HSV);
111
112     //Koboume otidhpote den einai kokkino
113     Core.inRange(matHSV, new Scalar(0, 100, 30), new Scalar(15, 255, 255),
          thresholded);
114     Core.inRange(matHSV, new Scalar(150, 100, 30), new Scalar(180, 255, 255),
          thresholded2);
115     Core.bitwise_or(thresholded, thresholded2, matFinal);
116
117     //filtraroume gia kalutera apotelesmata twn kuklwn
118     Imgproc.GaussianBlur(matFinal, matFinal, new Size(3, 3), 0);
119
120     // Variables for HoughCircles search
121     Mat circles = new Mat();
122     double minDist = matFinal.height();
123     int cannyUpperThreshold = 200;
124     int accumulatorThreshold = 20;
125     int minRadius = 15;
126     int maxRadius = 30;
127
128     Imgproc.HoughCircles(matFinal, circles, Imgproc.CV_HOUGH_GRADIENT, 1.5,
          minDist, cannyUpperThreshold, accumulatorThreshold, minRadius, maxRadius);
129
130     // if ball founded
131     if (circles.cols() > 0){
132       // retrieve ball coords from camera
133       double vCircle[] = circles.get(0,0);
134       double i = vCircle[0];
```

```
135     double j = vCircle[1];
136
137     // draw circle for founded ball
138     Point ptBall = new Point(Math.round(i), Math.round(j));
139     Core.circle(matFinal, ptBall, 25, new Scalar(255,255,255), 3);
140
141     // tranform from (0->240, 0->240) coord system to
142     // (-120->120, -120->120)
143     double x = i - matFinal.width()/2;
144     double y = -(j - matFinal.height()/2);
145     // tranform coords system, rotate 45 degrees
146     double newX = rotetedCoordX(x, y);
147     double newY = rotetedCoordY(x, y);
148     // for values 0-1 (max 135)
149     newX = newX/135;
150     newY = newY/135;
151
152     // compute time interval between 2 action
153     long systemNowTime = System.currentTimeMillis();
154     double timeIntervalMillis = (double) (systemNowTime - systemLastTime);
155     systemLastTime = systemNowTime;
156     timeInterval = timeIntervalMillis/1000;
157
158     // current error
159     ErrorCurX = desiredX - newX;
160     // integrate error
161     ErrorIntX = ErrorIntX + ErrorCurX * timeInterval;
162     // derivative error
163     double ErrorDerX = (ErrorCurX - ErrorPrevX)/timeInterval;
164     ErrorPrevX = ErrorCurX;
165     // calculate angle for rotate motor
166     int angleC = (int) Math.round((Kp*ErrorCurX) + (Ki*ErrorIntX) + (Kd*
            ErrorDerX));
167     Log.e("pid", "p " + Kp*ErrorCurX+" i " + Ki*ErrorIntX+" d " + Kd*ErrorDerX);
168     // cut large angles
169     if(angleC > maxAngle)
170       angleC = maxAngle;
171     else if(angleC < -maxAngle)
172       angleC = -maxAngle;
173
174     // rotate if not near to desired point
175       if((Math.abs(newX)-Math.abs(desiredX))>0.1){
176         motorC.rotateTo(-angleC, true);
177       }
178
179       // current error
180     ErrorCurY = desiredY - newY;
181     // integrate error
182     ErrorIntY = ErrorIntY + ErrorCurY * timeInterval;
183     // derivative error
184     double ErrorDerY = (ErrorCurY - ErrorPrevY)/timeInterval;
185     ErrorPrevY = ErrorCurY;
186     // calculate angle for rotate motor
187     int angleB = (int) Math.round((Kp*ErrorCurY) + (Ki*ErrorIntY) + (Kd*
            ErrorDerY));
188
189     if(angleB > maxAngle)
190       angleB = maxAngle;
191     else if(angleB < -maxAngle)
192       angleB = -maxAngle;
193
194     if((Math.abs(newY)-Math.abs(desiredY))>0.1){
195       motorB.rotateTo(-angleB, true);
196     }
197
198     if((Math.abs(newX)-Math.abs(desiredX))<0.1 && (Math.abs(newY)-Math.abs(
            desiredY))<0.1){
199       nxtCommands.playTone( 900, 10 );
200     }
201   }else{
```

```
202       // play one tone
203       nxtCommands.playTone( 784, 20 );
204       Log.e("detected", "no");
205     }
206
207     // draw desired position (in this cause center)
208     Point ptCenter = new Point(matFinal.width()/2, matFinal.height()/2);
209     Core.circle(matFinal, ptCenter, 3, new Scalar(255,255,255), 3);
210
211     // create bitmap image for output
212     Bitmap bmp = Bitmap.createBitmap(matFinal.width(), matFinal.height(), Bitmap.
            Config.ARGB_8888);
213
214     try {
215       // transform mat to bitmap and return to display
216       Utils.matToBitmap(matFinal, bmp);
217       return bmp;
218     } catch(Exception e) {
219       Log.e("AutoControlView", "fail in processFrame() " + e.getMessage());
220       bmp.recycle();
221       return null;
222     }
223   }
224
225   /**
226    * Rotate coordinate system 45 degrees
227    * compute new x
228    * @param x
229    * @param y
230    * @return new x
231    */
232   public double rotetedCoordX(double x, double y){
233     return Math.cos(Math.PI/4)*x - Math.sin(Math.PI/4)*y;
234   }
235
236   /**
237    * Rotate coordinate system 45 degrees
238    * compute new y
239    * @param x
240    * @param y
241    * @return new y
242    */
243   public double rotetedCoordY(double x, double y){
244     return Math.sin(Math.PI/4)*x + Math.cos(Math.PI/4)*y;
245   }
246
247   @Override
248   public void run() {
249     super.run();
250
251     synchronized (this) {
252       // Explicitly deallocate Mats
253       if (matHSV != null)
254          matHSV.release();
255       if (matRGB != null)
256          matRGB.release();
257       if (matFinal != null)
258          matFinal.release();
259       if (matRGBsq != null)
260          matRGBsq.release();
261       if (thresholded != null)
262         thresholded.release();
263       if (thresholded2 != null)
264         thresholded2.release();
265
266       matRGB = null;
267       matFinal = null;
268       matRGBsq = null;
269       matHSV = null;
270       thresholded = null;
```

```
271       thresholded2 = null;
272     }
273   }
274 }
```

AutoControlViewBase.java

```
1  package com.ucg.ballplateproject;
2
3  /**
4   * Ayto to kommati kwdika dinetai etoimo apo thn openCV.
5   * einai klash poy ry8mizei kai anoigei thn kamera
6   * stelnei tis plhrofories sthn klash AutoControlView
7   * kai lamvanei pisw to bitmap poy 8a emfanisei
8   *
9   * oi allages exoyn ginei stis diastaseis toy video
10  * kai h dior8wsh sthn emfanisei toy bitmap logw to oti
11  * exei epilegei to portrait orientation
12  */
13 import org.opencv.highgui.Highgui;
14 import org.opencv.highgui.VideoCapture;
15
16 import android.content.Context;
17 import android.graphics.Bitmap;
18 import android.graphics.Canvas;
19 import android.graphics.Matrix;
20 import android.util.Log;
21 import android.view.SurfaceHolder;
22 import android.view.SurfaceView;
23
24 public abstract class AutoControlViewBase extends SurfaceView implements
       SurfaceHolder.Callback, Runnable {
25   private static final String TAG = "AutoControlViewBase";
26
27   private SurfaceHolder mHolder;
28   private VideoCapture mCamera;
29
30   public AutoControlViewBase(Context context) {
31     super(context);
32     mHolder = getHolder();
33     mHolder.addCallback(this);
34   }
35
36   public boolean openCamera() {
37     synchronized (this) {
38       releaseCamera();
39       mCamera = new VideoCapture(Highgui.CV_CAP_ANDROID);
40       if (!mCamera.isOpened()) {
41         mCamera.release();
42         mCamera = null;
43         return false;
44       }
45     }
46     return true;
47   }
48
49   public void releaseCamera() {
50     synchronized (this) {
51       if (mCamera != null) {
52         mCamera.release();
53         mCamera = null;
54       }
55     }
56   }
57
58   public void setupCamera(int width, int height) {
59     synchronized (this) {
60       if (mCamera != null && mCamera.isOpened()) {
61         // desired resolution video
62         mCamera.set(Highgui.CV_CAP_PROP_FRAME_WIDTH, 320);
63         mCamera.set(Highgui.CV_CAP_PROP_FRAME_HEIGHT, 240);
64       }
65     }
66   }
67
```

```
68   public void surfaceChanged(SurfaceHolder _holder, int format, int width, int
         height) {
69     Log.i(TAG, "surfaceChanged");
70     setupCamera(width, height);
71   }
72
73   public void surfaceCreated(SurfaceHolder holder) {
74     Log.i(TAG, "surfaceCreated");
75     (new Thread(this)).start();
76   }
77
78   public void surfaceDestroyed(SurfaceHolder holder) {
79     Log.i(TAG, "surfaceDestroyed");
80     releaseCamera();
81   }
82
83   protected abstract Bitmap processFrame(VideoCapture capture);
84
85   public void run() {
86     while (true) {
87       Bitmap bmp = null;
88
89       synchronized (this) {
90         if (mCamera == null)
91           break;
92         if (!mCamera.grab())
93           break;
94
95         bmp = processFrame(mCamera);
96       }
97
98       if (bmp != null) {
99         Canvas canvas = mHolder.lockCanvas();
100        if (canvas != null) {
101          //Display Orientation Issue When Working with OpenCV (portrait)
102          Matrix matrix = new Matrix();
103          matrix.preTranslate((canvas.getWidth() - bmp.getWidth()) / 2, (canvas.
                 getHeight() - bmp.getHeight()) / 2);
104          matrix.postRotate(90f,(canvas.getWidth()) / 2,(canvas.getHeight()) / 2);
105          canvas.drawBitmap(bmp, matrix, null);
106          mHolder.unlockCanvasAndPost(canvas);
107        }
108        bmp.recycle();
109      }
110      try { Thread.sleep(100); } catch (InterruptedException e) {}
111    }
112  }
113 }
```

Main.java

```
1  package com.ucg.ballplateproject;
2
3  import java.io.IOException;
4  import java.io.InputStream;
5  import java.io.OutputStream;
6  import java.util.Iterator;
7  import java.util.Set;
8
9  import org.opencv.android.BaseLoaderCallback;
10 import org.opencv.android.LoaderCallbackInterface;
11 import org.opencv.android.OpenCVLoader;
12
13 import android.os.Bundle;
14 import android.app.Activity;
15 import android.bluetooth.BluetoothAdapter;
16 import android.bluetooth.BluetoothDevice;
17 import android.bluetooth.BluetoothSocket;
18 import android.content.BroadcastReceiver;
19 import android.content.Context;
20 import android.content.Intent;
21 import android.content.IntentFilter;
22 import android.hardware.Sensor;
23 import android.hardware.SensorEvent;
24 import android.hardware.SensorEventListener;
25 import android.hardware.SensorManager;
26 import android.util.Log;
27 import android.view.Window;
28 import android.widget.SeekBar;
29 import android.widget.SeekBar.OnSeekBarChangeListener;
30 import android.widget.TextView;
31 import android.widget.Toast;
32
33 public class Main extends Activity implements SensorEventListener,
       OnSeekBarChangeListener {
34
35   final String TAG = "Main";
36   final String ROBOTNAME = "NXT";
37
38   // modes
39   private int MODE_MANUAL_CONTROL = 0;
40   private int MODE_AUTO_CONTROL = 1;
41   private int MODE_LEVELING = 2;
42   private int mode;
43
44   // BT Variables
45   private BluetoothAdapter btAdapter;
46   private Set<BluetoothDevice> pairedDevices;
47   private BluetoothSocket socket;
48   private boolean bConnected = false;
49   // End BT Variables
50
51   // Sensor variables
52   private double sensorX = 0;
53   private double sensorY = 0;
54
55   // Settings variables (of two seekbars) for manual control
56   private int maxAngle = 0;
57   private int maxSpeed = 0;
58
59   // thread and thread flag for stopping thread
60   private runControl thread = null;
61   private boolean activeThread = false;
62
63   // Views
64   private TextView readings = null;
65   private TextView tvSpeed = null;
66   private TextView tvAngle = null;
67   private SeekBar sbSpeed = null;
```

```
68  private SeekBar sbAngle = null;
69
70  // class for set view (auto control mode)
71  private AutoControlView autoControlView;
72
73  // sensor manager
74  private SensorManager sManager = null;
75
76  // first time changed sensors
77  boolean first = true;
78
79  /**
80   * Called when the activity is first created. Inititializes all the
81   * graphical views.
82   */
83  @Override
84  public void onCreate(Bundle savedInstanceState) {
85     super.onCreate(savedInstanceState);
86     // Remove window title
87     getWindow().requestFeature(Window.FEATURE_NO_TITLE);
88
89     // Get info-extras from previous activity (Splash activity),
90     // in this case the user choice (manual, auto, leveling)
91     Bundle extras=getIntent().getExtras();
92
93     // For different user choice we set different content view
94     // (layout xml or class view) and initialize views like textviews, buttons
         ...
95     if(extras.getString("control").equals("manual")){
96     mode = MODE_MANUAL_CONTROL;
97     setContentView(R.layout.manual_control);
98
99     // initialize views
100    readings = (TextView)findViewById(R.id.tvSensorReadings);
101    tvSpeed = (TextView)findViewById(R.id.tvSpeed);
102    tvAngle = (TextView)findViewById(R.id.tvAngle);
103    sbSpeed = (SeekBar)findViewById(R.id.sbSpeed);
104    sbSpeed.setOnSeekBarChangeListener(this);
105    sbAngle = (SeekBar)findViewById(R.id.sbAngle);
106    sbAngle.setOnSeekBarChangeListener(this);
107
108    // initialize sensor manager
109    sManager = (SensorManager) getSystemService(SENSOR_SERVICE);
110    }else if(extras.getString("control").equals("auto")){
111     mode = MODE_AUTO_CONTROL;
112    }else if(extras.getString("control").equals("leveling")){
113    mode = MODE_LEVELING;
114    setContentView(R.layout.leveling);
115
116    // initialize view
117    readings = (TextView)findViewById(R.id.tvSensorReads);
118    // initialize sensor manager
119    sManager = (SensorManager) getSystemService(SENSOR_SERVICE);
120    }
121 }
122
123 /**
124  * Called when the activity is start after created or restarted.
125  * Set filters for receiver and try connect to robot
126  */
127 @Override
128 public void onStart(){
129   super.onStart();
130
131   // set filter for receiver, for detect when connected and disconnected with
         robot via bluetooth
132   registerReceiver(mReceiver,new IntentFilter("android.bluetooth.device.action.
         ACL_CONNECTED"));
133   registerReceiver(mReceiver,new IntentFilter("android.bluetooth.device.action.
         ACL_DISCONNECTED"));
```

```
134
135     findRobot();
136   }
137
138   /**
139    * Called when the activity is stopped.
140    * Disconnect robot
141    */
142   @Override
143   public void onStop() {
144     super.onStop();
145
146     activeThread = false;
147     try { Thread.sleep(500); } catch (InterruptedException e) {}
148     disconnectFromRobot();
149     unregisterReceiver(mReceiver);
150   }
151
152   /**
153    * Find specific device for connect and call function connect connectToRobot(
          BluetoothDevice)
154    */
155   public void findRobot(){
156     try{
157       // Get default bluetooth adapter
158       btAdapter = BluetoothAdapter.getDefaultAdapter();
159       // Get paired bluetooth devices
160       pairedDevices = btAdapter.getBondedDevices();
161       // set iterator over a sequence of bluetooth devices, like a list
162       Iterator<BluetoothDevice> it = pairedDevices.iterator();
163       // Scan the iterator/list for specific bluetooth device, the robot and call
            function to connect
164       while (it.hasNext()){
165         BluetoothDevice bd = it.next();
166
167         if (bd.getName().equalsIgnoreCase(ROBOTNAME)) {
168           connectToRobot(bd);
169           return;
170         }
171       }
172     }catch (Exception e){
173       Log.e(TAG,"Failed in findRobot() [" + e.getMessage() + "]");
174     }
175   }
176
177   /**
178    * Connect to robot socket
179    * @param Bluetooth Device for connect
180    */
181   private void connectToRobot(BluetoothDevice bd){
182     try{
183       socket = bd.createRfcommSocketToServiceRecord(java.util.UUID.fromString("
            00001101-0000-1000-8000-00805F9B34FB"));
184       socket.connect();
185       Toast.makeText(Main.this, "Robot Connected!", Toast.LENGTH_LONG).show();
186     }catch (Exception e){
187       Log.e(TAG,"Failed in connectToRobot() [" + e.getMessage() + "]");
188     }
189   }
190
191   /**
192    * Disconnect from robot socket
193    */
194   public void disconnectFromRobot(){
195     try{
196       socket.close();
197       Toast.makeText(Main.this, "Robot Disconnected!", Toast.LENGTH_LONG).show();
198     }catch (Exception e){
199       Log.e(TAG,"Failed in disconnectFromRobot() [" + e.getMessage() + "]");
200     }
```

```
201
202     if(mode == MODE_AUTO_CONTROL){
203       if (null != autoControlView)
204         autoControlView.releaseCamera();
205     }
206   }
207
208   /**
209    * Called when mobile and robot connected or disconnected (paired) via
           bluetooth
210    */
211   private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
212     @Override
213     public void onReceive(Context context,Intent intent) {
214       if (intent.getAction().equals("android.bluetooth.device.action.ACL_CONNECTED
              ")) {
215         handleConnected();
216       }
217       if (intent.getAction().equals("android.bluetooth.device.action.
              ACL_DISCONNECTED")) {
218         handleDisconnected();
219       }
220     }
221   };
222
223   /**
224    * Called when mobile and robot connected
225    */
226   private void handleConnected() {
227     try {
228       // If mode manual control or leveling start thread
229       if(mode == MODE_MANUAL_CONTROL || mode == MODE_LEVELING){
230         // Start orientation sensor
231         if (sManager != null) {
232           sManager.registerListener(Main.this,sManager.getDefaultSensor(Sensor.
                  TYPE_ORIENTATION), SensorManager.SENSOR_DELAY_UI);
233         }
234
235         thread = new runControl(socket.getInputStream(), socket.getOutputStream());
236       }else if(mode == MODE_AUTO_CONTROL){
237         if (!OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_2_4_2, this,
                openCVCallBack))
238         Log.e(TAG, "Cannot connect to OpenCV Manager");
239       }
240       bConnected = true;
241     } catch (Exception e) {
242       Log.e(TAG,"Failed in handleConnected() [" + e.getMessage() + "]");
243       disconnectFromRobot();
244     }
245   }
246
247   /**
248    * Called when mobile and robot disconnected and unregister sManager
249    */
250   private void handleDisconnected() {
251     bConnected = false;
252     if(mode == MODE_MANUAL_CONTROL || mode == MODE_LEVELING){
253       if (sManager != null) {
254         sManager.unregisterListener(Main.this);
255       }
256       readings.setText("Sensors Disabled. Please connect to Robot to resume.");
257     }
258   }
259
260
261   /**
262    * Called when mobile orientation changed
263    * Read sensor values
264    */
265   public void onSensorChanged(SensorEvent event) {
```

```
266     if (bConnected == false) return;
267
268     sensorX = event.values[1];
269     sensorY = event.values[2];
270     readings.setText("sensor X: " + sensorX + " sensor Y: " + sensorY);
271
272     // enable thread when started read sensor
273     if(first == true){
274       thread.start();
275       activeThread = true;
276       first = false;
277     }
278   }
279
280   public void onAccuracyChanged(Sensor s, int accuracy) {}
281
282
283   /**
284    * Thread for control robot in mode manual control and leveling
285    */
286   class runControl extends Thread{
287     private InputStream is = null;
288     private OutputStream os = null;
289
290     // Variables for check time interval between nxt commands
291     private long systemLastTime = System.currentTimeMillis() + 100;
292     private long elapsedSinceNXTCommand = 0;
293     public static final int UPDATE_TIME = 300;
294
295     // Sensor past coords values
296     private double lastX = 0;
297     private double lastY = 0;
298     private double lastSpeed = 0;
299
300     // Motors
301     public static final byte MOTOR_B = 0x01;
302     public static final byte MOTOR_C = 0x02;
303     NXTCommands nxtCommands;
304     Motor motorB = null;
305     Motor motorC = null;
306
307     // Lever arm offset (m)
308     public static final double D = 0.025;
309     // PLate size (center of the plate to point connecting lever arm)(m)
310     public static final double L = 0.045;
311
312     // class thread constructor
313     public runControl(InputStream inputStream, OutputStream outputStream){
314       this.is = inputStream;
315       this.os = outputStream;
316       // create new object for NXTCommands with is os of connected robot
317       nxtCommands = new NXTCommands(is, os);
318       // Play tone on nxt robot
319       nxtCommands.playTone( 784, 500 );
320
321       // Create objects for 2 motors, set streams of connected nxt,
322       // reset angle and set speed of motors
323       motorB = new Motor(MOTOR_B);
324       motorC = new Motor(MOTOR_C);
325       motorB.setStreams(is, os);
326       motorC.setStreams(is, os);
327       motorB.resetTachoCount();
328       motorC.resetTachoCount();
329       motorB.setSpeed(120);
330       motorC.setSpeed(120);
331     }
332
333     public void run() {
334       while (activeThread) {
335         updateTime();
```

```
336       //is it time to update motor movement?
337       if (elapsedSinceNXTCommand > UPDATE_TIME) {
338         elapsedSinceNXTCommand = 0;
339         if(mode == MODE_MANUAL_CONTROL && maxSpeed!= 0 && maxAngle!=0){
340           // tis mises moires tou sensora na strefetai o kinhthras
341           double newX = sensorX/2;
342           // tis mises moires tou sensora na strefetai o kinhthras,
343           // (-) gia na strefetai sthn antitheth pleyra
344           double newY = -sensorY/2;
345
346           // set limit angles
347           if(newX > maxAngle)
348             newX = maxAngle;
349           else if(newX < -maxAngle)
350             newX = -maxAngle;
351
352           if(newY > maxAngle)
353             newY = maxAngle;
354           else if(newY < -maxAngle)
355             newY = -maxAngle;
356
357           if(Math.abs(maxSpeed-lastSpeed) > 0 ){
358             motorB.setSpeed(maxSpeed);
359             motorC.setSpeed(maxSpeed);
360
361             lastSpeed = maxSpeed;
362           }
363           // angle changed
364           if(Math.abs(newX-lastX) > 0 ){
365             // set motor speed
366             // rotate to angle of the mobile (orientation)
367             motorB.rotateToReal((int)(newX), true);
368           }
369           if(Math.abs(newY-lastY) > 0 ){
370             // if changed orientation on x axis
371             motorC.rotateToReal((int)(newY), true);
372           }
373           lastX = newX;
374           lastY = newY;
375         }else if (mode == MODE_LEVELING){
376           double newX = sensorX;
377           double newY = sensorY;
378
379           // rotate to opposite direction
380           motorB.rotateToReal((int)(-Math.round(newX*L/D)), true);
381           motorC.rotateToReal((int)(Math.round(newY*L/D)), false);
382
383           // wait for the reset motors angles
384           try { Thread.sleep(100); } catch (InterruptedException e) {}
385           motorB.resetTachoCount();
386           motorC.resetTachoCount();
387         }
388       }
389     }
390   }
391
392   private void updateTime() {
393     long systemNowTime = System.currentTimeMillis();
394
395     // Do nothing if mLastTime is in the future.
396     // This allows to delay the start
397     if (systemLastTime > systemNowTime)
398       return;
399
400     long elapsed = systemNowTime - systemLastTime;
401     elapsedSinceNXTCommand += elapsed;
402     systemLastTime = systemNowTime;
403   }
404 }
405
```

```
406
407   /**
408    * Called when progress/seekbar changed and set maxSpeed and maxAngle values
409    */
410   public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser)
          {
411     switch(seekBar.getId()){
412     case R.id.sbSpeed:
413       // MaxSpeed range 30-130 and 0
414       if(progress==0)
415         maxSpeed = progress;
416       else
417         maxSpeed = progress+30;
418       tvSpeed.setText("Select max motors speed ("+maxSpeed+" degree/sec)");
419       break;
420     case R.id.sbAngle:
421       //MaxAngle range 0-25
422       maxAngle = progress/4;
423       tvAngle.setText("Select max motors angle ("+maxAngle+" degree)");
424       break;
425     }
426   }
427
428
429   public void onStartTrackingTouch(SeekBar seekBar) {}
430
431   public void onStopTrackingTouch(SeekBar seekBar) {}
432
433
434   /*********************************************
435    * FOR AUTO CONTROL - OPENCV CAMERA
436    *********************************************/
437   private BaseLoaderCallback openCVCallBack = new BaseLoaderCallback(this) {
438     @Override
439     public void onManagerConnected(int status) {
440       switch (status) {
441         case LoaderCallbackInterface.SUCCESS:
442           Log.i(TAG, "OpenCV loaded successfully");
443
444           try { // Create and set View
445             autoControlView = new AutoControlView(mAppContext, socket.getInputStream
                    (), socket.getOutputStream());
446           } catch (IOException e) {
447             Log.e(TAG,"Failed in onManagerConnected() [" + e.getMessage() + "]");
448           }
449           setContentView(autoControlView);
450           // Check native OpenCV camera
451           if( !autoControlView.openCamera() ) {
452             Toast.makeText(Main.this, "Can't open camera!", Toast.LENGTH_LONG).show
                    ();
453             finish();
454           }
455           break;
456         default:
457           super.onManagerConnected(status);
458           break;
459       }
460     }
461   };
462 }
```

Motor.java

```
1  package com.ucg.ballplateproject;
2
3  /**
4   * O kwdikas periexei kapoia kommatia apo tis entoles lejos
5   * (java entoles gia to NXT)
6   */
7
8  import java.io.InputStream;
9  import java.io.OutputStream;
10 import android.util.Log;
11
12 public class Motor {
13   private byte motor;
14   private byte power;
15   private byte mode;
16   private byte regulationMode;
17   private byte turnRatio;
18   private byte runState;
19
20   private NXTCommands nxtCommands;
21   private InputStream is = null;
22   private OutputStream os = null;
23   private int myTachoCount;
24   int countForUpdateTacho;
25
26   public Motor(byte motor){
27     this.motor = motor;
28     this.power = 80; // 80% power by default.
29     // Brake mode and regulation default
30     this.mode = (byte) (NXTProtocol.BRAKE + NXTProtocol.REGULATED);
31     this.regulationMode = NXTProtocol.REGULATION_MODE_IDLE;
32     this.turnRatio = 0;
33     this.runState = NXTProtocol.MOTOR_RUN_STATE_IDLE;
34     myTachoCount = 0;
35     countForUpdateTacho = 5;
36   }
37
38   public void setStreams(InputStream inputStream, OutputStream outputStream){
39     this.is = inputStream;
40     this.os = outputStream;
41     nxtCommands = new NXTCommands(is, os);
42   }
43
44   /**
45    * Sets motor speed , in degrees per second; Up to 900 is posssible
46    * with 8 volts. NOTE: If using LEGO firmware this will convert
47    * the number into power. 900 = 100% power, 450 = 50% power.
48    * @param speed value in degrees/sec
49    */
50   public void setSpeed(int speed) {
51     if (speed > 900 | speed < 0)
52       return;
53     speed = (speed * 100) / 900;
54     this.power = (byte) speed;
55   }
56
57   public int getSpeed() {
58     return (this.power * 900) / 100;
59   }
60
61   /**
62    * Returns the rotation count for the motor.
63    * @return Tachometer count.
64    */
65   public int getTachoCount() {
66     if(countForUpdateTacho == 0){
67       OutputState state = nxtCommands.getOutputState(motor);
68       countForUpdateTacho = 5;
```

```
69
70        Log.e("tacho error"," "+(state.tachoCount-myTachoCount));
71        myTachoCount = state.rotationCount;
72        return state.rotationCount;
73      }else{
74        countForUpdateTacho--;
75        return myTachoCount;
76      }
77    }
78
79    /**
80     * Returns the rotation count for the motor.
81     * (only real rotation)
82     * @return Tachometer count.
83     */
84    public int getTachoCountReal() {
85      OutputState state = nxtCommands.getOutputState(motor);
86      return state.rotationCount;
87    }
88
89    /**
90     * Causes the motor to rotate a certain count. The motor will also
91     * backtrack to the desired count when done. NOTE: This method returns
92     * almost immediately if returnNow = true.
93     * @param count Number of counts to rotate motor.
94     * @param returnNow When true, method returns before the rotation is complete.
95     * @return Error value. 0 means succcess.
96     */
97    public int rotate(long count, boolean returnNow) {
98      this.runState = NXTProtocol.MOTOR_RUN_STATE_RUNNING;
99
100     myTachoCount = myTachoCount + (int)count;
101     if(count == 0){
102       return 0;
103     }else if (count > 0)
104       nxtCommands.setOutputState(motor, power, this.mode + NXTProtocol.MOTORON,
              regulationMode, turnRatio, runState, (int) count);
105     else
106       nxtCommands.setOutputState(motor, (byte) -power, this.mode + NXTProtocol.
              MOTORON, regulationMode, turnRatio, runState, (int) -count);
107
108     if (returnNow)
109       return 0;
110     else {
111       // Check if mode is moving until done
112       while (isMoving()) {
113         Thread.yield();
114       }
115       return 0;
116     }
117   }
118
119   public boolean isMoving() {
120     OutputState os = nxtCommands.getOutputState(motor);
121     return os.runState != NXTProtocol.MOTOR_RUN_STATE_IDLE;
122   }
123
124   /**
125    * Rotates to a desired tacho count. Does not return until
126    * rotation done. Note: The tachocount can not be reset to zero.
127    * @param target
128    */
129   public int rotateTo(long target) {
130     return rotateTo(target, false);
131   }
132
133   /**
134    * Rotates to a desired tacho count. Returns before the rotation
135    * is done if you include true as the argument.
136    * @param target
```

```
137     */
138    public int rotateTo(long target, boolean returnNow) {
139      int tachometer = this.getTachoCount();
140      return rotate(target-tachometer, returnNow);
141    }
142
143    /**
144     * Rotates to a desired tacho count. Returns before the rotation is done if you
            include true as
145     * the argument.
146     * @param target
147     */
148    public int rotateToReal(long target, boolean returnNow) {
149      int tachometer = this.getTachoCountReal();
150      return rotate(target-tachometer, returnNow);
151    }
152
153    /**
154     * Resets the rotation counter to zero.
155     */
156    public void resetTachoCount() {
157      nxtCommands.resetMotorPosition(this.motor, false);
158      myTachoCount = 0;
159    }
160  }
```

NXTCommands.java

```
1  package com.ucg.ballplateproject;
2
3  /**
4   * O kwdikas periexei kapoia kommatia apo tis entoles lejos
5   * (java entoles gia to NXT)
6   */
7
8  import java.io.IOException;
9  import java.io.InputStream;
10 import java.io.OutputStream;
11
12 import android.util.Log;
13
14 public class NXTCommands implements NXTProtocol {
15   private InputStream is = null;
16   private OutputStream os = null;
17
18   // Constractor
19   public NXTCommands(InputStream inputStream, OutputStream outputStream){
20     this.is = inputStream;
21     this.os = outputStream;
22   }
23
24   /**
25    * @param port - Output port (0 - 2 or 0xFF for all three)
26    * @param power - Setpoint for power. (-100 to 100)
27    * @param mode - Setting the modes MOTORON, BRAKE, and/or REGULATED.
28    * This parameter is a bitfield, so to put it in brake mode and
29    * regulated, use BRAKEMODE + REGULATED
30    * @param regulationMode - see NXTProtocol for enumerations
31    * @param turnRatio - Need two motors? (-100 to 100)
32    * @param runState - see NXTProtocol for enumerations
33    * @param tachoLimit - Number of degrees(?) to rotate before stopping.
34    */ // ">>>" mh proshmasmenh dexia metatopish
35   public void setOutputState(int port, byte power, int mode, int regulationMode,
         int turnRatio, int runState, int tachoLimit) {
36     byte[] request = {
37       DIRECT_COMMAND_NOREPLY,
38       SET_OUTPUT_STATE,
39       (byte) port, power,
40       (byte) mode,
41       (byte) regulationMode,
42       (byte) turnRatio,
43       (byte) runState,
44       (byte) tachoLimit,
45       (byte) (tachoLimit >>> 8),
46       (byte) (tachoLimit >>> 16),
47       (byte) (tachoLimit >>> 24) };
48     sendData(request);
49   }
50
51   /**
52    * Retrieves the current output state for a port.
53    * @param port - 0 to 2
54    * @return OutputState - returns a container
55    * object for output state variables.
56    */
57   public OutputState getOutputState(byte port) {
58     byte[] request = { DIRECT_COMMAND_REPLY, GET_OUTPUT_STATE, port };
59     byte[] reply = sendData(request);
60
61     OutputState outputState = new OutputState(port);
62     outputState.status = reply[4];
63     outputState.outputPort = reply[5];
64     outputState.power = reply[6];
65     outputState.mode = reply[7];
66     outputState.regulationMode = reply[8];
67     outputState.turnRatio = reply[9];
```

```
68    outputState.runState = reply[10];
69    outputState.tachoLimit = (0xFF & reply[11]) | ((0xFF & reply[12]) << 8) | ((0
          xFF & reply[13]) << 16) | ((0xFF & reply[14]) << 24);
70    outputState.tachoCount = (0xFF & reply[15]) | ((0xFF & reply[16]) << 8) | ((0
          xFF & reply[17]) << 16) | ((0xFF & reply[18]) << 24);
71    outputState.blockTachoCount = (0xFF & reply[19]) | ((0xFF & reply[20]) << 8) |
           ((0xFF & reply[21]) << 16) | ((0xFF & reply[22]) << 24);
72    outputState.rotationCount = (0xFF & reply[23]) | ((0xFF & reply[24]) << 8) |
          ((0xFF & reply[25]) << 16) | ((0xFF & reply[26]) << 24);
73    return outputState;
74  }
75
76  /**
77   * Resets either RotationCount or BlockTacho
78   * @param port Output port (0-2)
79   * @param relative TRUE: BlockTacho, FALSE: RotationCount
80   */
81  public void resetMotorPosition(int port, boolean relative) {
82    byte boolVal = 0;
83    if (relative)
84      boolVal = (byte) 0xFF;
85    byte[] request = { DIRECT_COMMAND_NOREPLY, RESET_MOTOR_POSITION, (byte) port,
          boolVal };
86    sendData(request);
87  }
88
89  /**
90   * Plays a tone on NXT speaker. If a new tone is sent while the previous
91   * tone is playing, the new tone command will stop the old tone command.
92   * @param frequency - 100 to 2000?
93   * @param duration - In milliseconds.
94   * @return - Returns true if command worked, false if it failed.
95   */
96  public void playTone(int frequency, int duration) {
97    byte[] request = {
98      DIRECT_COMMAND_NOREPLY,
99      PLAY_TONE,
100     (byte) frequency,
101     (byte) (frequency >>> 8),
102     (byte) duration,
103     (byte) (duration >>> 8) };
104   sendData(request);
105 }
106
107 /**
108  * Method to read response data
109  * @return response from nxt (with LSB, MSB)
110  */
111 public byte[] readData() {
112   // max 64 Bytes +LSB,MSB
113   byte[] buffer = new byte[66];
114   int numBytes;
115
116   try {
117     numBytes = is.read(buffer);
118   } catch (IOException e) {
119     Log.e("readData", "Read failed.", e);
120     throw new RuntimeException(e);
121   }
122
123   byte[] result = new byte[numBytes];
124
125   for (int i = 0; i < numBytes; i++) {
126     result[i] = buffer[i];
127   }
128
129   return result;
130 }
131
```

```
132   /**
133    * Method to send request to NXT and return verification result.
134    * @param request to nxt (without LSB, MSB)
135    * @return response from nxt (with LSB, MSB)
136    */
137   public byte[] sendData(byte[] request) {
138     int lsb = request.length;
139     int msb = request.length >>> 8;
140     byte verify[] = null;
141
142     try {
143       Thread.sleep(30);
144       os.write((byte) lsb);
145       os.write((byte) msb);
146       os.write(request);
147     } catch (IOException e) {
148       Log.e("sendData", "Write failed.", e);
149       throw new RuntimeException(e);
150     } catch (InterruptedException e) {}
151
152     if(request[0] == DIRECT_COMMAND_REPLY){
153       byte[] reply = readData();
154       verify = reply;
155     }
156
157     return verify;
158   }
159 }
```

NXTProtocol.java

```
1  package com.ucg.ballplateproject;
2
3  public interface NXTProtocol {
4    /**
5     * NXT Protocol - LCP Commands (only needed)
6     */
7    // Command types constants. Indicates type of packet being sent or received.
8    public static byte DIRECT_COMMAND_REPLY = 0x00;
9    public static byte REPLY_COMMAND = 0x02;
10   public static byte DIRECT_COMMAND_NOREPLY = (byte) 0x80; // Avoids 100ms
         latency
11
12   // Direct Commands
13   public static byte PLAY_TONE = 0x03;
14   public static byte SET_OUTPUT_STATE = 0x04;
15   public static byte GET_OUTPUT_STATE = 0x06;
16   public static byte RESET_MOTOR_POSITION = 0x0A;
17
18   // Output state constants
19   // Mode:
20   public static byte MOTORON = 0x01; //Turn on the specified motor
21   public static byte BRAKE = 0x02; //Use run/brake instead of run/float in PWM
22   public static byte REGULATED = 0x04;//Turns on the regulation
23
24   // Regulation Mode:
25   public static byte REGULATION_MODE_IDLE = 0x00;   //No regulation will be
         enabled
26   public static byte REGULATION_MODE_MOTOR_SPEED = 0x01; //Power control will be
         enabled on specified output
27   public static byte REGULATION_MODE_MOTOR_SYNC = 0x02; //Synchronization will be
          enabled (Needs enabled on two output)
28
29   // RunState:
30   public static byte MOTOR_RUN_STATE_IDLE = 0x00;  //Output will be idle
31   public static byte MOTOR_RUN_STATE_RAMPUP = 0x10; //Output will ramp-up
32   public static byte MOTOR_RUN_STATE_RUNNING = 0x20; //Output will be running
33   public static byte MOTOR_RUN_STATE_RAMPDOWN = 0x40; //Output will ramp-down
34 }
```

OutputState.java

```
1  package com.ucg.ballplateproject;
2
3  public class OutputState {
4    // Status of the NXTCommand.getOutputState command.
5    public byte status;
6    // (Range: 0 to 2)
7    public byte outputPort;
8    // -100 to 100
9    public byte power;
10   public byte mode;
11   public byte regulationMode;
12   // -100 to 100
13   public byte turnRatio;
14   public byte runState;
15   // Current limit on a movement in progress, if any
16   public long tachoLimit;
17   // Internal count. Number of counts since last reset of the motor counter
18   public int tachoCount;
19   // Current position relative to last programmed movement
20   public int blockTachoCount;
21   // Current position relative to last reset of the rotation sensor for motor
22   public int rotationCount;
23
24   public OutputState(int port) {
25     outputPort = (byte) port;
26   }
27 }
```

Splash.java

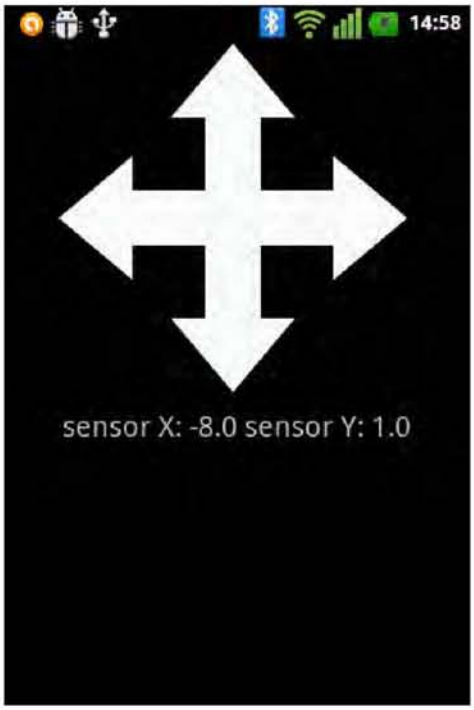
```
1  package com.ucg.ballplateproject;
2
3  import android.app.Activity;
4  import android.content.Intent;
5  import android.os.Bundle;
6  import android.view.View;
7  import android.view.Window;
8  import android.view.View.OnClickListener;
9  import android.widget.Button;
10
11 public class Splash extends Activity implements OnClickListener{
12
13   private Button bAutoControl;
14   private Button bManualControl;
15   private Button bAbout;
16   private Button bLeveling;
17
18   @Override
19   protected void onCreate(Bundle savedInstanceState) {
20     super.onCreate(savedInstanceState);
21     // Remove window title
22     getWindow().requestFeature(Window.FEATURE_NO_TITLE);
23
24     setContentView(R.layout.splash);
25
26     // Initialize buttons
27     bLeveling = (Button)findViewById(R.id.bLeveling);
28     bLeveling.setOnClickListener(this);
29     bAutoControl = (Button)findViewById(R.id.bAutoControl);
30     bAutoControl.setOnClickListener(this);
31     bManualControl = (Button)findViewById(R.id.bManualControl);
32     bManualControl.setOnClickListener(this);
33     bAbout = (Button)findViewById(R.id.bAbout);
34     bAbout.setOnClickListener(this);
35   }
36
37   // Set buttons events
38   public void onClick(View v) {
39     switch(v.getId()){
40     case R.id.bAbout:
41       Intent intentAbout = new Intent(this, About.class);
42       startActivity(intentAbout);
43       break;
44     case R.id.bAutoControl:
45       Intent intentMainA = new Intent(this, Main.class);
46       intentMainA.putExtra("control", "auto");
47       startActivity(intentMainA);
48       break;
49     case R.id.bManualControl:
50       Intent intentMainM = new Intent(this, Main.class);
51       intentMainM.putExtra("control", "manual");
52       startActivity(intentMainM);
53       break;
54     case R.id.bLeveling:
55       Intent intentMainL = new Intent(this, Main.class);
56       intentMainL.putExtra("control", "leveling");
57       startActivity(intentMainL);
58       break;
59     }
60   }
61 }
```

Παράρτημα Γ΄

Γραφικό περιβάλλον χρήστη της εφαρμογής

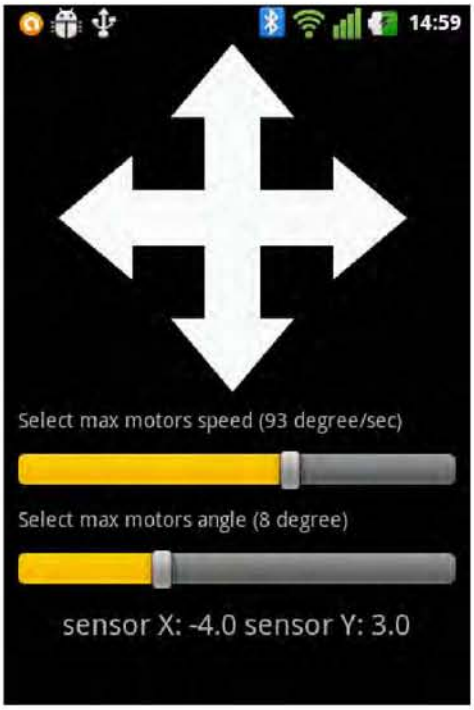
Το γραφικό περιβάλλον χρήστη (Graphical User Interface - GUI) της εφαρμογής που δημιουργήθηκε για το σύστημα αυτόματου ελέγχου σφαίρας σε πλάκα.

Σχήμα Γ΄.1: *Κεντρικό Activity*



Σχήμα Γ΄.2: *Λειτουργία Leveling*

Σχήμα Γ.3: *Λειτουργία Auto Control (Camera)*



Σχήμα Γ.4: *Λειτουργία Manual Control (Accelerometer)*

Σχήμα Γ.5: *Σχετικά με την εφαρμογή*

Παράρτημα Δ΄

Τεχνικά χαρακτηριστικά του Lego Mindstorms

Το σετ Lego Mindstorms NXT 2.0 αποτελείται από τα IntelligentBrick, 4 αισθητήρες, 3 σερβο-κινητήρες, 7 καλώδια RJ12, και 600 διάφορα κομμάτια Lego.

Επεξεργαστής	ARM7, 32 bit
RAM	64 KB
Μνήμη	FLASH, 256 KB
Μικροελεγκτής	AVR , 8 bit
Οθόνη	LCD, 100 x 64 pixels, ασπρόμαυρη
Communication	Bluetooth και καλώδιο USB

Πίνακας Δ΄.1: *Τεχνικά χαρακτηριστικά του IntelligentBrick*

Επιπλέον έχει 4 κουμπιά για αλληλεπίδραση με το brick, ένα ηχείο, 4 θύρες εισόδου για τους αισθητήρες και 3 θύρες εξόδου για τους σερβο-κινητήρες. Οι αισθητήρες και οι σερβο-κινητήρες επικοινωνούν με το brick με καλώδια RJ12, τα οποία είναι παρόμοια με τα καλώδια τηλεφώνου RJ11 . Οι αισθητήρες είναι οι εξής: ένας αισθητήρας υπερήχων (Ultrasonic Sensor), ένας αισθητήρας χρώματος (Color Sensor) και δύο αισθητήρες αφής (Touch Sensors)[47, 48].

Παράρτημα Ε΄

Τεχνικά χαρακτηριστικά της κινητής συσκευής Smartphone

Λειτουργικό Σύστημα	Android OS, v2.3.3 (Froyo)
Chipset	Qualcomm MSM7227
CPU	ARM 11, 600 MHz
GPU	Adreno 200
RAM	512 MB
Sensors	Accelerometer, proximity, compass
Οθόνη	οθόνη αφής TFT, 256K colors, 320 x 480 pixels, 3.2 inches
Camera	3.15 MP
Video	VGA, 18fps
Bluetooth	v2.1 with A2DP
Διαστάσεις	113.5 x 59 x 13.3 mm
Βάρος	129 g

Πίνακας Ε΄.1: *Τεχνικά χαρακτηρίστηκα της κινητής συσκευής Smartphone [49]*

Βιβλιογραφία

[1] *http://el.wikipedia.org/wiki/%CE%A1%CE%BF%CE%BC%CF%80%CF%8C%CF%84*.

[2] *http://www.britannica.com/EBchecked/topic/505818/robot*.

[3] *http://en.wikipedia.org/wiki/Robot*.

[4] Stuart J. Russell and Peter Norvig. *Τεχνητή Νοημοσύνη Μια σύγχρονη προσέγγιση*. Εκδόοεις Κλειδάριθμος, 2005.

[5] Ιωάννης Βλαχάβας, Πέτρος Κεφαλάς, Νικόλαος Βασιλειάδης, Φώτης Κόκκορας, and Ηλίας Σακελλαρίου. *Τεχνητή Νοημοσύνη Γ' Έκδοση*. Εκδόοεις Πανεπιστημίου Μακεδονίας, 2006.

[6] *http://en.wikipedia.org/wiki/ASIMO*.

[7] *http://ja.wikipedia.org/wiki/AIBO*.

[8] *https://en.wikipedia.org/wiki/Mars_Exploration_Rover*.

[9] *http://jimtsutsui.wordpress.com/2011/05/14/microscopic-robotic-surgery/*.

[10] Richard C. Dorf and Robert H. Bishop. *Σύγχρονα Συστήματα Αυτόματου Ελέγχου*. Εκδόοεις Τζιόλα, 2009.

[11] Νικόλαος-Δημήτριος Τανταρούδας. Γραμμικοποίηση εισόδου - κατάστασης και εισόδου - εξόδου μη γραμμικού συστήματος οφαίρας-ράβδου. Master's thesis, Πανεπιστήμιο Πατρών, τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών, 2011.

[12] Wei Wang. Control of a ball and beam system. Master's thesis, University of Adelaide, School of Mechanical Engineering, 2007.

[13] Elif Kocak. Control law partitioning applied to beam and ball problem. Master's thesis, Middle East Technical University, Graduate School of Natural and Applied Sciences, 2008.

[14] *http://aboushehri.mech.iut.ac.ir/content/control-beam-and-ball-system*.

[15] Jonothan Bruce, Chris Keeling, and Ronald Rodriguez. Four degree of freedom control system using a ball on a plate. Master's thesis, Southern Polytechnic State University, 2011. Bachelor's thesis.

[16] Magnus Espersson. Vision algorithms for ball on beam and plate. Master's thesis, Lund University, Department of Automatic Control, 2010.

[17] Phillip Yip. Symbol-based control of a ball-on-plate mechanical system. Master's thesis, University of Maryland, 2004.

[18] *https://sites.google.com/site/controlandelectronics/b/*.

[19] *http://ctms.engin.umich.edu/CTMS/index.php?example=BallBeam&section=SystemModeling*.

[20] *http://en.wikipedia.org/wiki/Lagrangian*.

[21] *http://el.wikipedia.org/wiki/ÑïðÞ_áäñÜíåéáò*.

[22] *http://gun.teipir.gr/DSAELAB/Ergastiriakes/pidtutorial.pdf*.

[23] *http://en.wikipedia.org/wiki/PID_controller*.

[24] *http://en.wikipedia.org/wiki/Lego_Mindstorms*.

[25] *http://www.pestaola.gr/4-must-see-lego-mindstorms-nxt-videos/*.

[26] *http://www.whodiesinthenewharrypotterbook.com/lego-games/lego-mindstorms-nxt-2-0-parts-list/*.

[27] *http://edurobotics.weebly.com/*.

[28] *http://mindstorms.lego.com/en-us/whatisnxt/default.aspx*.

[29] *http://en.wikipedia.org/wiki/Android_(operating_system)*.

[30] *http://en.wikipedia.org/wiki/List_of_features_in_Androi)*.

[31] Wei-Meng Lee. *Beginning Android Application Development.* Wiley Publishing, 2011.

[32] *http://el.wikipedia.org/wiki/Android.*

[33] *http://en.wikipedia.org/wiki/Android_version_history.*

[34] *http://www.knowahead.in/dyk/androids-versions-are-named-in-alphabetical-order/.*

[35] *http://en.wikipedia.org/wiki/OpenCV.*

[36] urlhttp://opencv.org/about.html.

[37] Lego Mindstorms. *Lego Mindstorms NXT Bluetooth Developer Kit*, 2006.

[38] Lego Mindstorms. *Lego Mindstorms NXT Direct Commands*, 2006.

[39] *http://en.wikipedia.org/wiki/Eclipse_(software).*

[40] *http://developer.android.com/tools/sdk/eclipse-adt.html.*

[41] *http://developer.android.com/guide.*

[42] W.Frank Ableson, Robi Sen, Chris King, and C.Enrique Ortiz. *Android in Action Third Edition.* Manning, 2012.

[43] *http://en.wikipedia.org/wiki/Bluetooth_protocols.*

[44] *http://mindstorms.lego.com/en-us/news/readmore/default.aspx?id=227417.*

[45] *http://www.codeproject.com/Articles/508676/Breaking-Out-of-The-Box-An-Introduction-to-BlackBe.*

[46] *http://en.wikipedia.org/wiki/Rotation_matrix.*

[47] *http://shop.lego.com/en-US/LEGO-MINDSTORMS-NXT-2-0-8547.*

[48] *http://en.wikipedia.org/wiki/Lego_Mindstorms_NXT#NXT_Intelligent_Brick.*

[49] *http://www.gsmarena.com/lg_optimus_one_p500-3516.php.*