Πανεπιστήμιο Θεσσαλίας
Σχολή Θετικών Επιστημών
Τμήμα Πληροφορικής με Εφαρμογές στη Βιοϊατρική

Πτυχιακή Εργασία

Τίτλος: Κατασκευή και ευφυής έλεγχος ρομπότ

Εκπόνηση: Μπαρμπούνης Παντελής

Α.Μ: 2007020

Επίβλεψη: Πλαγιανάκος Βασίλειος

Λαμία 2014

Ευχαριστήριο σημείωμα

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, Πλαγιανάκο Βασίλειο, για την καθοδήγηση του. Επίσης θα ήθελα να ευχαριστήσω την οικογένειά μου και τους φίλους μου για τη βοήθεια τους και που ήταν δίπλα μου καθ'όλη τη διάρκεια της σταδιοδρομίας μου.

Abstract

In this thesis, is presented a robotic construction, by using the educational robot Lego Mindstorms NXT, made by Lego. Its purpose is to patrol on points that we have introduced, through a graphical user interface on a personal computer. This robot is a commercial product and its form is not predisposed. On the contrary, it can take whatever form we want it to, either by following the instruction set that comes with the package, or from our imagination. The whole set sontains a lot of plastic parts just like the classic Lego bricks, as well as other accessories such as, gears, sensors and motors. With these parts we created a vehicle that can move back and forth and turn. On this vehicle, were adjusted an ultrasonic sensor and a sound sensor.

Our robot is called to follow the line, which we drew on the computer, and return from the same path, as many times as we want. So essentially it patrols. Firstly, the robot receives the data from the computer via USB or Bluetooth and awaits the sound to start, which in this case is the clap of our hands. It is following the path and in the meanwhile the ultrasonic sensor is "on guard" looking out for obstacles and once it detects one; it is maneuvering to overtake it. Whenever we want, we clap our hands again and the robot stop moving.

Future changes could be, adding a camera with the ability to record data or even send them through a wireless communication via Bluetooth. Moreover, it would be possible to add more sensors to gather more information, e.g. an infrared sensor. Tablet computers and android devices could be also used, giving us the ability to draw the path directly with our finger on the touch screen.

Περίληψη

Στην πτυχιακή αυτή, παρουσιάζεται μία ρομποτική κατασκευή, με τη χρήση του εκπαιδευτικού ρομπότ της Lego, το Lego Mindstorms NXT. Σκοπός της είναι να περιπολεί πάνω σε σημεία που έχουμε εισάγει, μέσω γραφικού περιβάλλοντος στον υπολογιστή. Το εκπαιδευτικό ρομπότ αυτό αποτελεί εμπορικό προϊόν και δεν έχει προδιατεταγμένη μορφή. Αντίθετα, μπορεί να πάρει τη μορφή που θέλουμε, είτε ακολουθώντας τις οδηγείες που έρχονται μαζί με το πακέτο, είτε με τη φαντασία μας. Στη συσκευασία περιλαμβάνονται πολλά πλαστικά μέρη, όπως τα κλασσικά τουβλάκια της Lego, καθώς εξαρτήματα όπως γρανάζια, αισθητήρες και κινητήρες. Με αυτά τα κομμάτια κατασκευάστηκε ένα όχημα, το οποίο μπορεί να κινείται μπρος-πίσω και να στρίβει. Σε αυτό προσαρμόστηκαν, ένας αισθητήρας υπερήχων καθώς και ένας αισθητήρας ήχου.

Το ρομπότ μας καλείται να ακολουθήσει τη γραμμή, που θα σχεδιάσουμε στον υπολογιστή, καθώς και να επιστρέψει από το ίδιο μονοπάτι, όσες φορές θέλουμε. Κάνει στην ουσία ένα είδος περιπολείας. Αρχικά λαμβάνει τα δεδομένα από τον υπολογιστή μέσω USB ή Bluetooth και αναμένει να λάβει τον ήχο της έναρξης, ο οποίος στην προκειμένη περίπτωση είναι ο κρότος των χεριών μας. Ακολουθώντας το μονοπάτι, ο αισθητήρας απόστασης βρίσκεται σε «επιφυλακή» προσέχοντας για εμπόδια και μόλις τα εντοπίσει κάνει ελιγμό για να τα αποφύγει. Όποτε θελήσουμε ξαναχτυπάμε τα χέρια μας και το ρομπότ σταματάει να κινείται.

Μελλοντικές αλλαγές θα μπορούσαν να είναι, η προσθήκη κάμερας με δυνατότητα καταγραφής των δεδομένων ή ακόμα και αποστολή των δεδομένων της, μέσω ασύρματης επικοινωνίας Bluetooth. Ακόμη θα ήταν εφικτή και η προσθήκη επιπλέον αισθητήρων για τη συλλογή περισσότερων δεδομένων, π.χ. αισθητήρας υπερύθρων. Επίσης θα μπορούσαν να χρησιμοποιηθούν tablets και άλλες συσκευές android όπως για παράδειγμα smart-phones και να ζωγραφίζουμε απευθείας με το δάκτυλό μας στην οθόνη αφής.

Περιεχόμενα

1. ΕΙΣΑΓΩΓΗ 7

1.1 Το θέμα της πτυχιακής 7

1.2 Περιγγαφή των κεφαλαίων 7

2. ΡΟΜΠΟΤΙΚΗ 9

2.1 Τι είναι ρομπότ 9

2.2 Τι είναι ρομποτική 9

2.3 Μυθολογική και ιστορική αναδρομή 9

2.4 Είδη ρομπότ και εφαρμογές τους 11

2.5 Ρομπότ στην καθημερινή ζωή και εργασία 12

3. LEGO ΚΑΙ LEGO MINDSTORMS 13

3.1 Σύντομη ιστορική αναδρομή της LEGO 13

3.2 Ιστορική ανδρομή της LEGO MINDSTORMS 13

3.3 Το φυσικό υλικό του ρομπότ 14

3.4 Γλώσσες προγραμματισμού για το NXT 16

4. ΚΑΤΑΣΚΕΥΗ ΤΟΥ ΡΟΜΠΟΤ 19

4.1 Σχεδιατικές αποφάσεις 19

4.1.1 Αισθητήρας ήχου 19

4.1.2 Αισθητήρας απόστασης 19

4.2 Η τελική κατασκευή 19

5. ΤΟ ΓΡΑΦΙΚΟ ΠΕΡΙΒΑΛΛΟΝ ΧΡΗΣΤΗ (GUI) 36

5.1 Τα μέρη που αποτελούν το γραφικό περιβάλλον (GUI).... 36

5.2 Αναλυτική λειτουργία των πλήκτρων..... 38

6. ΤΟ ΜΟΝΟΠΑΤΙ 40

6.1 Τι είναι σημείο στο επίπεδο..... 40

6.2 Υπολογισμός ευθύγραμμου τμήματος.....40

6.3 Υπολογισμός γωνίας μεταξύ δύο ευθύγραμμων τμημάτων 42

6.4 Υπολογισμός της γωνίας σε περιστροφές ρόδας.....44

7. ΚΙΝΗΣΗ ΤΟΥ ΡΟΜΠΟΤ 46

7.1 Λήψη δεδομένων..... 46

7.2 Ξεκινώντας τα δύο νήματα (threads) 46

7.3 Έλεγχος για εμπόδιο 47

7.4 Ακολουθώντας το μονοπάτι.....47

8. ΕΜΠΟΔΙΟ ΣΤΟ ΜΟΝΟΠΑΤΙ 49

8.1 Εμπόδιο εντοπίστηκε 49

8.2 Η συνάρτηση overTake() 49

8.2.1 Η πρώτη περίπτωση 50

8.2.2 Η δεύτερη περίπτωση 51

8.2.3 Η τρίτη περίπτωση 51

9. ΣΥΜΠΕΡΑΣΜΑΤΑ 50

9.1 Προβλήματα – Περιορισμοί.....51

9.2 Βελτιώσεις 51

9.3 Μελλοντικές χρήσεις..... 52

ΒΙΒΛΙΟΓΡΑΦΙΑ 53

ΠΑΡΑΡΤΗΜΑ Α 54

ΠΑΡΑΡΤΗΜΑ Β 90

1. ΕΙΣΑΓΩΓΗ

1.1 Το θέμα της πτυχιακής εργασίας

Στην πτυχιακή αυτή, παρουσιάζεται μία ρομποτική κατασκευή, η οποία ακολουθεί σημεία τα οποία έχουμε εισάγει στον υπολογιστή μέσω γραφικού περιβάλλοντος και φτάνοντας στο τέλος τους επιστρέφει. Αυτή τη διαδικασία την επαναλαμβάνει όσες φορές θελήσουμε. Το ρομπότ μας είναι κατασκευασμένο με κομμάτια που θυμίζουν τα κλασσικά τουβλάκια της Lego, συνδυασμένα με αισθητήρες και έναν μικρό επεξεργαστή.

Στον υπολογιστή εμφανίζεται το γραφικό μας περιβάλλον, στο οποίο, ζωγραφίζουμε το μονοπάτι που επιθυμούμε και στη συνέχεια στέλνουμε τα δεδομένα στο ρομπότ μας, μέσω USB ή Bluetooth. Μόλις τα λάβει, περιμένει τον ήχο έναρξης, ο οποίος είναι το χτύπημα των χεριών μας, και ξεκινάει να ακολουθεί το μονοπάτι που σχεδιάσαμε. Φτάνοντας στο τέλος, επιστρέφει μέσω της ίδιας διαδρομής και ούτω καθεξής. Καθ' όλη τη διάρκεια ελέγχει αν στο δρόμο του συναντήσει εμπόδια ή απότομο τέλος του δαπέδου.

1.2 Περιγραφή κεφαλαίων

Παρακάτω βλέπουμε τα κεφάλαια και το περιεχόμενά τους,

Κεφάλαιο 1:

Παρουσιάζεται το θέμα της πτυχιακής και δίνεται μια περιγραφή των κεφαλαίων.

Κεφάλαιο 2:

Δίνεται ο ορισμός του ρομπότ και της ρομποτικής, μια ιστορική αναδρομή, καθώς και εφαρμογές τους στην καθημερινότητα.

Κεφάλαιο 3:

Ιστορική αναδρομή της Lego και της πλατφόρμα Lego Mindstorms. Αναλυτική περιγραφή του επεξεργαστή, των αισθητήρων, των σερβοκινητήρων και των δομικών στοιχείων που την απαρτίζουν. Ακόμη, γίνεται μία μικρή αναφορά σε ορισμένες γλώσσες προγραμματισμού για το Lego Mindstorms NXT.

Κεφάλαιο 4:

Δίνεται σε βήματα η κατασκευή του ρομπότ, με εικόνες, καθώς και ο λόγος για την επιλογή της τελικής μορφής.

Κεφάλαιο 5:

Σε αυτό το κεφάλαιο, γίνεται λόγος για το γραφικό περιβάλλον του προγράμματος και αναλύονται και επεξηγούνται τα συστατικά του μέρη.

Κεφάλαιο 6:

Γίνεται αναλυτική περιγραφή του τρόπου σκέψης, για τον υπολογισμό του μονοπατιού που ακολούθεί το ρομπότ μας.

Κεφάλαιο 7:

Σε αυτό το κεφάλαιο, περιγράφονται τα πράγματα από την οπτική γωνία του ρομπότ στο κομμάτι της πλοήγησης. Ορισμένοι τρόποι που σκεφτήκαμε για την ακολουθία του μονοπατιού και που απορρίψαμε, καθώς και ο τελικός τρόπος που επιλέξαμε.

Κεφάλαιο 8:

Εδώ θα αναλυθεί ο τρόπος σκέψης και ο κώδικας που χρησιμοποιήσαμε για να αποφύγουμε το τυχόν εμπόδιο.

Κεφάλαιο 9:

Παραθέτονται προβλήματα που αντιμετωπίσαμε, καθώς και συμπεράσματα στα οποία καταλήξαμε.

Βιβλιογραφία:

Δίνονται οι πηγές από τις οποίες αντλήσαμε πληροφορίες για την πτυχιακή, αυτή, εργασία.

Παράρτημα:

Δίνεται ο τρόπος εγκατάστασης των κατάλληλων οδηγών για τη λειτουργία του LeJOS και του NXT καθώς και ο κώδικας στο κομμάτι του υπολογιστή και στο κομμάτι του ρομπότ με αναλυτικά σχόλια και επισημάνσεις.

2. ΡΟΜΠΟΤΙΚΗ

2.1 Τι είναι ρομποτική

Ως ρομποτική χαρακτηρίζουμε τον σύγχρονο τεχνολογικό κλάδο της αυτοματοποίησης. Δίνει έμφαση στο σχεδιασμό και τη λειτουργία τους καθώς και στη μελέτη των ρομπότ όπως επίσης και στην περεταίρω έρευνα για βελτιώσεις υπαρχόντων ρομποτικών κατασκευών. (1)

2.2 Τι είναι το ρομπότ

Σύμφωνα με τον ορισμό του Ινστιτούτου Ρομπότ των ΗΠΑ, «*ρομπότ είναι μια επαναπρογραμματιζόμενη πολυλειτουργική χειριστική διάταξη, σχεδιασμένη για τη μετακίνηση υλικών, εξαρτημάτων, εργαλείων και εξειδικευμένων διατάξεων, μέσω μεταβλητών, προγραμματισμένων κινήσεων για την εκτέλεση μιας σειράς εργασιών*». Ένα ρομπότ συγκροτείται από δύο συστήματα, το ***μηχανικό*** και το ***ηλεκτρονικό***. Στο πρώτο ανήκει το σύστημα κίνησης και στο δεύτερο υπάγεται η επαναπρογραμματιζόμενη μνήμη του. Οι τροποι με τους οποίους μπορούμε να διαχωρίσουμε τις αντίστοιχες κατηγορίες των ρομπότ οφείλονται σε διάφορα κριτίρια. Ένα από αυτά είναι η διάκρισή τους σε τρεις "γενιές". Τα ρομπότ **πρώτης γενιάς** ανήκουν αυτά τα οποία αποκλειστικά διευθύνονται από τον άνθρωπο και έχουν περιορισμένη ευελιξία. Για παράδειγμα, οι λεγόμενοι "χειριστές", οι οποίοι είναι σχετικά απλά εργαλεία που δίνουν τη δυνατότητα στον διυθύνων, τη μετακίνηση επικίνδυνων αντικειμένων (π.χ. ραδιενεργών υλικών). Στη **δεύτερη γενιά** κατατάσσονται τα ρομπότ που λειτουργούν βάση ενός σταθερού προγράμματος δράσης και εκείνα που λαμβάνουν εντολές από κάποιο σύστημα αριθμητικού ελέγχου. Τέλος, στην **τρίτη γενιά** κατατάσσονται ρομπότ που έχουν τη δυνατότητα:

- να λάβουν αισθητήριες "πληροφορίες" από το περιβάλλον,
- να επεξεργαστούν τις πληροφορίες αυτές
- να εκτελέσουν εργασίες με τη βοήθεια κινητήριου συστήματος. (1)

2.3 Μυθολογική και ιστορική αναδρομή

Το πρώτο ρομπότ στην ιστορία ήταν ο Τάλως. Ο μυθικός χάλκινος γίγαντας που προστάτευε την μινωική Κρήτη από κάθε επίδοξο εισβολέα. Ο Τάλως είναι από τις

πιο αγαπητές μυθικές προσωπικότητες του αρχαίου κόσμου και ένας από τους πιο σημαντικούς ελληνικούς μύθους.

Ο Τάλως δεν γεννήθηκε αλλά δημιουργήθηκε και υπάρχουν τρεις παραδοχές στο μύθο για τη δημιουργία του. Η μία αναφέρεται στον ίδιο το Δία ως δημιουργό του, η δεύτερη στον πολυτεχνίτη Δαίδαλο ο οποίος ενέργησε με εντολή του Δία και η τρίτη στον Ήφαιστο, θεό της φωτιάς και του σιδήρου. Ο βασιλιάς των θεών, Διας, έκανε τρία δώρα προς την Ευρώπη, τον Τάλω, ένα χρυσό σκύλο που δεν του ξεύφευγε κανένα θήραμα και μία φαρέτρα με βέλη που δεν αστοχούσαν ποτέ. Η Ευρώπη έκανε έκανε τρεις γιούς, το Μίνωα, μυθικό βασιλιά της Κνωσού στον οποίο μετά χάρισε τον Τάλω, τον Ραδάμανθυ και τον Σαρπηδόνα. (2)

Φυσικά, όπως προαναφέραμε, ο Τάλως ανήκει στην ελληνική μυθολογία. Η κατασκευή του **πρώτου** ρομπότ αποδίδεται στον μαθηματικό **Αρχύτα τον Ταραντίνο** (428-347 π.Χ.), ο οποίος λέγεται πως κατασκεύασε μια ιπτάμενη μηχανή τη λεγόμενη "περιστερά" ή "πετομηχανή". Αυτή η μηχανή ήταν ικανή να διανύσει έως και 200 μέτρα απόσταση και κινούταν με ατμό. Περίπου το **150 π.Χ.** είχε κατασκευαστεί ο Μηχανισμός των Αντικυθήρων, ο αρχαιότερος αυτοματισμός, που είχε τη δυνατότητα να εντοπίσει τις θέσεις των πλανητών. Ο **Ήρων ο Αλεξανδρεύς** περίπου το **10 – 70 μ.Χ.** κατασκεύασε ένα αυτοκινούμενο τρίκυκλο. Το πρώτο ανθρωποειδές ρομπότ, έναν προγραμματιζόμενο τυμπανιστή, κατασκεύασε ο Άραβας **Al-Jazari.** Το **1495** ο Ιταλός **Leonardo DaVinci** σχεδιάζει και πιθανόν κατασκευάζει το παλαιότερο σχέδιο ανθρωποειδούς ρομπότ που σώζεται μέχρι σήμερα. Πρόκειται για μια μηχανική κατασκευή που μοιάζει με στρατιώτη που φοράει πανοπλία της εποχής. Ο μηχανισμός στο εσωτερικό του, του έδινε τη δυνατότητα να κινεί τα χέρια και το κεφάλι του και να ανασηκώνεται, όπως ακριβώς θα γινόταν εάν βρισκόταν ένας άνθρωπος μέσα στην πανοπλία. Το **1738** ο **Jacques de Vaucanson** ξεκίνησε να κατασκευάζει ρομπότ. Η αρχική κατασκευή ήταν ένας μηχανισμός που παίζει φλάουτο και μπορεί να παίξει 12 τραγούδια. Στην συνέχεια την επέκτεινε κατασκευάζοντας ένα ρομπότ που μπορούσε να παίξει φλάουτο και τύμπανο. Η πιο γνωστή όμως κατασκευή του ήταν μια ρομποτική πάπια που είχε τη δυνατότητα να κινείται, να κουνάει τα φτερά της ακόμα και να τρώει σπόρους. Η πάπια αποτελούσε ένα παράδειγμα «κινούμενης ανατομίας», όπως την αποκαλούσε αυτό το κομμάτι της μελέτης του, δηλαδή της μοντελοποίησης της ανατομίας ανθρώπων και ζώων. Το **1898** ο Σέρβος **Nikola Tesla** παρουσιάζει το πρώτο τηλεχειριζόμενο πλοίο. Το **1930** η εταιρία **Westinghouse Electric Corporation** στην Η.Π.Α. δημμιουργεί ένα ανθρωποειδές ρομπότ, με το όνομα Electro που μπορούσε να μιλάει, να περπαταέι και να καπνίζει. Το **1936** ο **Alan Turing** εισάγει την έννοια των μηχανών υπολογισμού, γνωστές ως μηχανές Turing. Αργότερα θα δημοσιεύσει το "Υπολογιστικές Μηχανές και Νοημοσύνη" (Computing Machinery and Intelligence), όπου θα προτείνει ένα πείραμα (Turing test) που ορίζει για το πότε μια μηχανή μπορεί να καλείται νοήμων. Το **1942** ο συγγραφέας επιστημονικής φαντασίας **Issac Asimov**, στο χρονογράφημά του *Runaround*, εισάγει τη λέξη *Robotics* και ορίζει ως ρομπότ ένα αυτόματο με εμφάνιση ανθρώπου αλλά απαλλαγμένο από συναισθήματα.

Η μεγαλύτερη συνεισφορά του Asimov, πέρα από την εισαγωγή της έννοιας της Ρομποτικής, είναι η δημιουργία των τριών βασικών νόμων των ρομπότ:

- ***Νόμος 1:*** Ένα ρομπότ δεν πρέπει ποτέ να βλάψει ένα άνθρωπο ή λόγω αδράνειάς του να αφήσει ένα άνθρωπο να πάθει κακό.

- ***Νόμος 2:*** Ένα ρομπότ πρέπει πάντοτε να υπακούει τις εντολές που δίνονται από τους ανθρώπους εκτός και αν συγκρούονται με ανώτερο νόμο.

- ***Νόμος 3:*** Ένα ρομπότ πρέπει να προστατέψει την ύπαρξή του εκτός και αν αυτό συγκρούεται με ένα ανώτερο νόμο. (4)

Το **1948** στο **πανεπιστήμιο του Bristol** στην Αγγλία κατασκευάζεται το πρώτο αυτόνομο ρομπότ με το όνομα Elsie, το οποίο λάμβανε ερεθίσμάτα με τη βοήθεια αισθητήρων φωτός και κινούταν με βάση αυτά. Το **1962** ο **George Devol** κατασκευάζει το πρώτο σύγχρονο, ψηφιακά προγραμματιζόμενο ρομποτικό βραχίονα με το όνομα Unimate, το οποίο είναι σχεδιασμένο να φέρνει σε πέρας επικίνδυνες, δύσκολες ή επαναλαμβανόμενες και ανιαρές εργασίες. Το **1966** το **Ινστιτούτο Έρευνας του πανεπιστημίου** του Stanford κατασκευάζει το Shakey, το πρώτο κινούμενο ρομπότ με δυνατότητες Τεχνητής Νοημοσύνης. Ο Shakey μπορούσε να εντοπίσει οπτικά αντικείμενα και προχωρήσει προς αυτά καθώς επίσης και να αλληλεπιδράσει με αυτα, όπως για παράδειγμα να τα σπρώξει. Το **1980** ο **Seymour Papert** δημοσιεύει το βιβλίο "Mindstorms: Children, Computers, and Powerful Ideas" όπου προτείνει για πρώτη φορά τη χρήση εκπαιδευτικών ρομπότ και εισάγει την έννοια της εκμάθησης μέσα από την κατασκευή και το πείραμα. Το **1986** η συνεργασία της **LEGO με το MIT** οδηγεί στην κυκλοφορία του πρώτου εκπαιδευτικού προϊόντος βασισμένο σε Lego, το LEGO tc Logo. Ο προγραμματισμός του γινόταν σε Logo και χρησιμοποιήθηκε ευρέως στην εκπαιδευτική διαδικασία και ειδικά στην διδασκαλία εννοιών της επιστήμης των υπολογιστών. (3) (5) (6)

2.4 Είδη ρομπότ και εφαρμογές

Τα είδη των ρομπότ θα μπορούσε εύκολα κανείς να πει πως είναι όσες και οι εφαρμογές για τις οποίες καλείται μια ομάδα μηχανικών να κατασκευάσει ένα ρομπότ. Ρομπότ συναντώνται σε αποστολές του στρατού για εξερεύνηση επικίνδυνων περιοχών, στην αστυνομία για εξουδετέρωση βομβών, στη βιομηχανία απαντώνται σε πολλές εφαρμογές πάνω στη γραμμή παραγωγής, στα διαστημικά προγράμματα για την εξερεύνηση του διαστήματος, στα μουσεία για την ξενάγηση των επισκεπτών, στην ιατρική χειρουργική, στην εκπαίδευση με τη χρήση εκπαιδευτικών ρομπότ όπως το NXT της LEGO, αλλά και αλλού.

Τα ρομπότ μπορούν να φτιαχτούν για να επιτελούν επαναλαμβανόμενες λειτουργίες είτε όχι. Μπορούν να φτιαχτούν ώστε να μοιάζουν με άνθρωπο (androids) ή με έντομα και να τα μιμούνται είτε να μοιάζουν με έναν βραχίονα και να επιτελούν μια μόνο εργασία πχ να κάνουν συγκόλληση 13 μετάλλων στη γραμμή

παραγωγής αυτοκινήτων. Υπάρχουν ρομπότ αμαξίδια που δουλεία τους είναι η εξερεύνηση περιοχών. Και πολλά ακόμη.

Η χρήση των ρομπότ γίνεται επιβεβλημένη όταν απαιτείται μεγάλη ακρίβεια, αυξημένη παραγωγικότητα και μακράς διάρκειας εργασία καθώς ο ανθρώπινος οργανισμός δε μπορεί να αντεπεξέλθει σε αυτές τις απαιτήσεις πέραν ενός σημείου. Επίσης θα ήταν ανήθικο και αντιδεοντολογικό. Πάντα όμως η ανθρώπινη νόηση (ακόμα τουλάχιστον) θα είναι απαραίτητη για την εύρωστη λειτουργία και έλεγχο των ρομποτικών μηχανών.

2.5 <u>Ρομπότ στην καθημερινή ζωή και εργασία</u>

Στο επίπεδο της καθημερινής ζωής τα ρομπότ είναι κατά κόρον μηχανικές συσκευές που είναι προγραμματισμένες να εκτελούν συγκεκριμένες επαναλαμβανόμενες λειτουργίες, να χρησιμοποιούνται για εργασίες επικίνδυνες ή δύσκολα πραγματοποιήσιμες από τον άνθρωπο, καθώς εργασίες μέσα στο σπίτι. Έτσι, υπάρχουν ρομπότ ικανά να καθαρίσουν το σπίτι, να μαγειρέψουν ή να μας διασκεδάσουν. Οι ρομποτικές συσκευές χρησιμοποιούνται συνήθως για την εκτέλεση πολλών εργασιών, που οι άνθρωποι είτε δεν μπορούν να κάνουν, επειδή είναι ιδιαιτέρως πολύπλοκες, είτε δεν θέλουν, επειδή είναι βαρετές, βρώμικες ή επικίνδυνες. Ένα παράδειγμα ρομποτικών εφαρμογών που έχουμε σήμερα βρίσκεται στην κατασκευή και συναρμολόγηση των αυτοκινήτων. Τα ρομπότ παίρνουν τη θέση των εργαζομένων στη γραμμή συναρμολόγησης των εργοστασίων, όπου εκτελούνται εξειδικευμένες εργασίες, όπως η τοποθέτηση καρφιών, η συναρμολόγηση βαρέων εξαρτημάτων, η βαφή κ.λπ.. Άλλο ένα παράδειγμα, πιο κοντά στην καθημερινότητα του μέσου ανθρώπου είναι το πλύσιμο ρούχων με τη χρήση πλυντηρίου. Το πληντύριο είναι και αυτό ένα είδος ρομπότ που έχει προγραμματιστεί έτσι ώστε παίρνοντας ως είσοδο την επιλογή προγράμματος πλύσης από τον άνθρωπο, ‘γνωρίζει’ σε τι βαθμούς να ζεστάνει το νερό, πόσες φορές να στίψει τα ρούχα και σε πόση ώρα να σταματήσει. Επίσης σε εχθρικά περιβάλλοντα, όπως τα ηφαίστεια, μελετώνται με τη χρήση ρομπότ, τα οποία ελέγχονται εξ αποστάσεως από ανθρώπους για τη συλλογή περιβαλλοντικών δειγμάτων του εδάφους, λάβα και μαγματικά υλικά.

Σύμφωνα με τον καθηγητή Hiroshi Ishiguro, ο οποίος έχει αφιερωθεί στη δημιουργία ανθρωπόμορφων ρομπότ, που δύσκολα διακρίνονται από τους πραγματικούς ανθρώπους*: «Η στιγμή που τα ρομπότ θα είναι τόσο κοινά όσο τα αυτοκίνητα – και μάλιστα φθηνότερα – είναι πολύ κοντά.»*

3. LEGO και LEGO MINDSTORMS

3.1 Σύντομη ιστορική αναδρομή της Lego

Η Lego είναι μία δημοφιλής γραμμή παιχνιδιών κατασκευής, που κατασκευάζονται από το Lego Group, μια ιδιωτική εταιρία με έδρα το Billund, στη Διανία. Ναυαρχίδα της εταιρίας αποτελούν τα πολύχρωμα πλαστικά τουβλάκια που συνδέονται μεταξύ τους, καθώς και οι συνοδευτικές σειρές με γρανάζια, μινιατούρες και διάφορα άλλα κομμάτια. Τα τουβλάκια Lego μπορούν να συναρμολογηθούν και να συνδεθούν με πολλούς τρόπους, με σκοπό την κατασκευή οχημάτων, κτηρίων, ακόμη και **κινούμενων ρομπότ**. Οτιδήποτε δημιουργείται μπορεί να αποσυναρμολογηθεί ώστε να κατασκευαστούν με τα κομμάτια, άλλες κατασκευές. Ο Όμιλος Lego ξεκίνησε στο εργαστήριο του **Ole Kirk Christiansen** (γεννήθηκε **7 Απρίλη 1891**), ένας ξυλουργός από το Billund, Δανία, ο οποίος άρχισε να κάνει ξύλινα παιχνίδια το **1932**. Το **1934**, η εταιρεία του ήρθε να κληθεί "Lego", από τη δανική φράση "leg godt", που σημαίνει "παίξτε καλά" ή "καλό παιχνίδι". (7)

3.2 Ιστορική αναδρομή της Lego Mindstorms

Τα Lego Mindstorms αποτελούν μια σειρά προϊόντων της Lego η οποία αποτελείται από προγραμματιζόμενα "τούβλα" (programmable bricks), κινητήρες, αισθητήρες, απλά τουβλάκια Lego και διάφορα άλλα μηχανικά μέρη, όπως άξονες, ακτίνες και λοιπά εργαλεία. Προσφέρει στον χρήστη τη δυνατότητα κατασκευής μοντέλων ενσωματωμένων συστημάτων, αποτελούμενων από ηλεκτρομηχανικά μέρη, τα οποία ο χρήστης ελέγχει μέσω υπολογιστή. Χρησιμοποιώντας τα Mindstorms, έχουμε τη δυνατότητα να διαμορφώσουμε πολλά είδη πραγματικών ενσωματωμένων συστημάτων, από ελεγκτές ανελκυστήρων έως βιομηχανικά ρομπότ.

Η ιστορία της πλατφόρμας Mindstorms ξεκινά όταν η Lego χρηματοδότησε μέρος της έρευνάς του MIT και έτσι αναπτύχθηκε στο Media Lab η πρώτη έκδοση του προγραμματιζόμενου "τούβλου" (programmable brick). Η πρώτη έκδοση Lego Mindstorms κυκλοφόρησε από από την Lego το 1998 κάτω από την επωνυμία Robotics Invention System (RIS). Αποτελούνταν από 717 κομμάτια τα οποία περιελάμβαναν απλά τουβλάκια Lego, κινητήρες, αισθητήρες καθώς και διάφορα άλλα μηχανικά μέρη, όπως άξονες, ακτίνες και ένα "RCX Brick" με ενσωματωμένο μικροεπεξεργαστή που προγραμματίζεται και αποτελεί τον "εγκέφαλο" του συστήματος. Η πρώτη αυτή έκδοση Lego Mindstorms έδωσε στους χρήστες τη δυνατότητα να κατασκευάσουν και να προγραμματίσουν μια μεγάλη ποικιλία από μοντέλα ενσωματωμένων συστημάτων. Το προϊόν, γνώρισε μεγάλη εμπορική επιτυχία με αποτέλεσμα να αναπτυχθούν και επιπλέον προγραμματιστικά περιβάλλοντα για το RCX brick όπως για παράδειγμα το LegoOS και το, βασισμένο σε Java, TinyVM. (8) (9)

Η επόμενη έκδοση Lego Mindstorms κυκλοφόρησε στην αγορά το 2006 με το όνομα Lego Mindstorms NXT το οποίο βασίζεται στο επιτυχημένο Robotics System Invention της εταιρείας, το οποίο έχει βελτιωθεί με την πρόσθεση νέων τεχνολογιών και αισθητήρων αυξημένων ικανοτήτων, ενώ το 2009 κυκλοφόρησε και η δεύτερη έκδοση του NXT, Lego Mindstorms NXT 2.0. Παράλληλα, κυκλοφορεί και εκπαιδευτική έκδοση, καλούμενη Lego Mindstorms for Schools η οποία χρησιμοποιεί το ROBOLAB, ένα περιβάλλον προγραμματισμού βασισμένο με γραφική διεπαφή, που αναπτύχθηκε στο Πανεπιστήμιο Tufts χρησιμοποιώντας ως μηχανή το National Instruments LabVIEW.

3.3 Το φυσικό υλικό του Ρομπότ

Κεντρική μονάδα

Η κεντρική μονάδα του NXT (NXT brick) είναι ο "εγκέφαλος" του ρομπότ. Μπορεί να προγραμματιστεί έτσι ώστε το ρομποτ να παίρνει αποφάσεις και να εκτελεί έξυπνες ενέργειες. Οι τεχνικές προδιαγραφές, της κεντρικής μονάδας, είναι οι ακόλουθες:

- Μικροεπεξεργαστής 32-bit ARM7
- 64 Kbytes RAM
- 256 Kbytes FLASH
- Μικροεπεξεργαστής 8-bit AVR
- 4 Kbytes FLASH
- 512 Byte RAM
- Bluetooth ασύρματη επικοινωνία (Class II V2.0)
- Θύρα USB 2.0 (12 Mbit/s)
- Τέσσερις θύρες εισόδου(ψηφιακή πλατφόρμα 6 καλωδίων και η μια θύρα περιλαμβάνει θύρα επέκτασης IEC 61158 Type4/EN 50 170 για μελλοντική χρήση)
- Τρείς θύρες εξόδου (ψηφιακή πλατφόρμα 6 καλωδίων)
- Οθόνη LCD 60x100 εικονοστοιχείων
- Ηχείο 8 Ohm (Ωμ)
- Πηγή ενέργειας: επαναφορτιζόμενη μπαταρία λιθίου ή 6 μπαταρίες AA
- Βύσμα τροφοδοσίας: US: 120 VAC 60Hz/ UK, EU, AUS: 230 ~ 50Hz

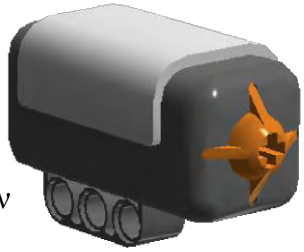
Μοτέρ κίνησης

Τα τρία μοτέρ παρέχουν στο ρομπότ την ικανότητα να κινηθεί ή να κινήσει αντικείμενα. Τα μοτέρ ονομάζονται αλληλεπιδρώντα (interactive) γιατί έχουν ενσωματωμένους αισθητήρες περιστροφής και ανατροφοδοτούν την κεντρική μονάδα με την τιμή περιστροφής του κάθε άξονα. Η δυνατότητα αυτή επιτρέπει τον έλεγχο των κινήσεων με μεγάλη ακρίβεια. Ο ενσωματωμένος αισθητήρας περιστροφών, μετράει τις περιστροφές της μηχανής σε μοίρες (με ακρίβεια +/- 1 μοίρα) ή σε πλήρεις περιστροφές. Μια περιστροφή είναι 360 μοίρες, άρα εαν προγραμματιστεί η μηχανή να στρίψει 180 μοίρες, τότε ο άξονας περιστροφής του κινητήρα θα πραγματοποιήσει μισή περιστροφή.

Λαμπτήρες

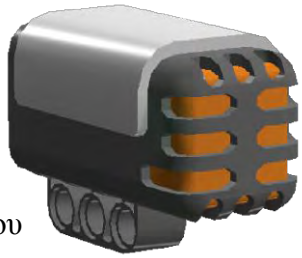
Οι λαμπτήρες μπορούν να ανάβουν και να σβήνουν, δημιουργώντας μοτίβα λάμψης φωτός. Μπορούν επίσης να χρησιμοποιηθούν για να ενεργοποιήσουν τον αισθητήρα φωτός ή για να δείξουν ότι κάποιο μοτέρ είναι σε λειτουργία ή ακόμη για να δείξουν την κατάσταση ενός αισθητήρα. Μπορούν επίσης να χρησιμοποιηθούν για να ζωντανέψουν τα «μάτια» του ρομπότ ή όπως αλλιώς κρίνεται απαραίτητο.

Αισθητήρες



Αφής: Ο ασθητήρας αφής μπορεί να προστεθεί σε ένα ρομπότ NXT και στη συνέχεια να προγραμματιστεί η συμπεριφορά του μοντέλου έτσι ώστε να διαφοροποιείται όταν ο αισθητήρας αφής πιέζεται ή απελευθερώνεται.

Ήχου: Ο αισθητήρας ήχου ανιχνεύει το επίπεδο των θορύβων και επιστρέφει την τιμή σε μονάδες προσαρμοσμένων ντεσιμπέλ (%dB): την υψηλή ή τη χαμηλή ένταση ήχου. Ο αισθητήρας ήχου μπορεί να μετρήσει σε dBA που αφορά στους ήχους που μπορούν να γίνουν αισθητοί από το ανθρώπινο αυτί και σεdB που αφορά όλους τους ήχους, συμπεριλαμβανομένων αυτών που είναι πολύ δυνατοί ή πολύ χαμηλοί για να ακουστούν απο ανθρώπους. Ο αισθητήρας ήχου μπορεί να μετρήσει επίπεδα ήχου μέχρι 90 dB. Οι ενδείξεις του αισθητήρα ήχου στο NXT εκφράζονται σε ποσοστό (%) του ήχου που ο αισθητήρας είναι ικανός να ανιχνεύσει. Για παράδειγμα, 4-5% είναι το επίπεδο ήχου σε ένα δωμάτιο όπου επικρατεί ησυχία και 5-10% είναι περίπου το επίπεδου ήχου

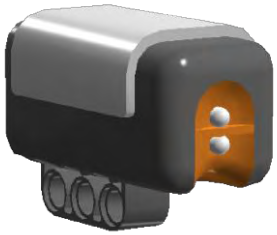


κάποιου που μιλάει από κάποια απόσταση. Μία ήρεμη συζήτηση ή η απαλή μουσική, κοντά στον αισθητήρα, κυμαίνεται μεταξύ 10 και 30%, ενώ το 30-100% αντιπροσωπεύει παραδείγματα όπως δυνατές φωνές ανθρώπων ή πολύ δυνατή μουσική.
Αυτές οι κλίμακες προϋποθέτουν μια απόσταση περίπου ενός μέτρου μεταξύ της πηγής ήχου και του αισθητήρα ήχου.

Υπερήχων

Ο αισθητήρας υπερήχων επιτρέπει στο ρομπότ να διακρίνει και να αναγνωρίζει αντικείμενα, να αποφεύγει εμπόδια, να μετράει αποστάσεις και να ανιχνεύει κινήσεις. Ο αισθητήρας υπερήχων στηρίζεται στην ίδια επιστημονική αρχή με τις νυχτερίδες. Μετράει την απόσταση υπολογίζοντας τον χρόνο που χρειάζεται ένα κύμα ήχου να χτυπήσει ένα αντικείμενο και να επιστρέψει, όπως συμπαίνει με την ηχώ. Ο αισθητήρας αυτός, μετράει την απόσταση σε εκατοστά ή ίντσες. Μπορεί να μετρήσει αποστάσεις από 0 έως 2,5 μέτρα με ακρίβεια +/- 3 εκατοστά. Τα αντικείμενα με μεγάλο μέγεθος και με σκληρές επιφάνειες παρέχουν τα καλύτερα αποτελέσματα. Αντικείμενα που είναι φτιαγμένα από μαλακό ύφασμα, κυρτά αντικείμενα (π.χ. μία μπάλα) ή πολύ λεπτά και μικρά αντικείμενα, μπορεί να δυσκολέψουν τον αισθητήρα να τα εντοπίσει.

Φωτός



Ο αισθητήρας φωτός επιτρέπει στο ρομπότ να διακρίνει τη διαφορά μεταξύ φωτός και σκοταδιού, να διακρίνει την ένταση του φωτός στον χώρο και να μετρήσει την ένταση του φωτός σε έγχρωμες επιφάνειες.

Μνήμη

Κάθε NXT έχει διαθέσιμη μνήμη περίπου 130,7 Kbytes. Ένα μέρος αυτής της μνήμης έχει ήδη χρησιμοποιηθεί για να αποθηκευτούν παραδείγματα προγραμμάτων, γραφικά και αρχεία ήχων. Το αποτέλεσμα είναι να απομένουν περίπου 56 Kbytes ελεύθερα για τα αρχεία που θα δημιουργηθούν από το χρήστη. (10)

3.4 Γλώσσες προγραμματισμού για το NXT

Το ρομπότ NXT, προγραμματίζεται με όλες σχεδόν τις γνωστές γλώσσες προγραμματισμού (C, C++, Java, .Net, κ.α) αν και η LEGO έχει φροντίσει να εκδώσει μία εκπαιδευτική γλώσσα οπτικού προγραμματισμού για το NXT [LEGO Mindstorms Edu NXT Software] σε συνεργασία με την εταιρία ανάπτυξης λογισμικού National Instruments (το εκπαιδευτικό λογισμικό που αναπτύχθηκε

αποτελεί μια εκπαιδευτική προσαρμογή του LabView και χρησιμοποιούν τον ίδιο compiler -G-). Παρακάτω δίνονται μερικές λεπτομέρειες για τις κυριότερες γλώσσες προγραμματισμού:

NXT-G
Ο προγραμματισμός γίνεται πολύ εύκολα με την "οπτική" γλώσσα που διαθέτει η Lego για τα Mindstorms NXT. Ακόμα και αν δεν έχετε καθόλου γνώσεις προγραμματισμού, μετά από μερικά λεπτά εξοικείωσης θα είστε σε θέση να γράψετε ακόμα και πολύ σύνθετα προγράμματα. Έχει ένα ευχάριστο γραφικό περιβάλλον, που είναι ιδανικό για εκπαιδευτικούς σκοπούς, αλλά η απλότητά του, τα μεγάλα σε όγκο εκτελέσιμα που δημιουργεί, αλλά και το ότι το LabView είναι σαν περιβάλλον πολύ βαρύ, ακόμα και σε σύγχρονους υπολογιστές, το καθιστά ανεπαρκές, οπότε, προτείνονται άλλες γλώσσες. (11)

Not eXactly C (NXC)
Η NXC όπως δηλώνει το όνομα της είναι μια γλώσσα υψηλού επιπέδου, παρόμοια της C βασισμένη στον μεταγλωτιστή της NBC. Για να μεταγλωτίσουμε NXC προγράμματα χρησιμοποιούμε τον μεταγλωτιστή της NBC μαζί με τα αρχεία του πηγαίου κώδικα που έχουν την επέκταση «.nxc» . Ως γλώσσα είναι απλή, έχει καλή υποστήριξη, δημιουργεί γρήγορα τα εκτελέσιμα και προτείνεται για σοβαρές υλοποιήσεις. Ένα μειονέκτημα της είναι η μη υποστήριξη επικοινωνίας μέσω Bluetooth, ανάμεσα σε ένα NXT και κάποια άλλη συσκευή. Υποστηρίζεται μονάχα, επικοινωνία μεταξύ 2 ή περισσοτέρων NXT. (12)

leJOS NXT
Το leJOS NXT είναι περιβάλλον προγραμματισμού σε Java για το LEGO MINDSTORMS NXT το οποίο και θα χρησιμοποιήσουμε. Μας επιτρέπει να προγραμματίσουμε το ρομπότ σε Java. Περιέχει:

- Firmware για το NXT που συμπεριλαμβάνει μια Java Virtual Machine
- Βιβλιοθήκες από κλάσεις Java που εφαρμόζουν το leJOS NXT API και προσφέρει μια εναλλακτική Java Runtime που είναι βελτιστοποιημένη για το NXT
- Ένα συνδετήρα για τη σύνδεση κλάσεων Java με το classes.jar για να σχηματίσουν ένα δυαδικό αρχείο που μπορεί να μεταφορτωθεί και να τρέξει στο NXT
- Εργαλεία για τον υπολογιστή που βοηθούν στην εγκατάσταση του firmware, στη μεταφόρτωση προγραμμάτων, τον εντοπισμό σφαλμάτων και πολλές άλλες λειτουργίες.
- Ένα API του υπολογιστή για τη σύνταξη προγραμμάτων που επικοινωνούν με προγράμματα leJOS NXT, χρησιμοποιώντας Bluetooth ή USB, ή χρησιμοποιώντας το πρωτόκολο επικοινωνίας της LEGO (LEGO Communication Protocol(LCP))
- Πολλά παραδείγματα

Μερικά από τα πλεονεκτήματά του είναι: Πολύ γρηγορότερη από την NXT-G , έχει υψηλού επιπέδου ανοιχτό κώδικα, υποστηρίζει πλήρως την σύνδεση μέσω Bluetooth και χρησιμοποιεί την στάνταρ γλώσσα προγραμματισμού Java. (13)

MATLAB και Simulink
Το MATLAB είναι ευρέως διαδεδομένο πρόγραμμα και χρησιμοποιείται για επιστημονικούς υπολογισμούς και ανάλυση δεδομένων και μπορεί επίσης να χρησιμοποιηθεί για τον έλεγχο του NXT χρησιμοποιώντας σύνδεση Bluetooth ή USB. Το Simulink είναι ένα περιβάλλον που στηρίζεται σε MATLAB και χρησιμοποιείται για την προσομοίωση και μοντελοποίηση δυναμικών συστημάτων. Με το Simulink μπορούν να σχεδιαστούν αλγόριθμοι ελέγχου του NXT χρησιμοποιώντας την γλώσσα προγραμματισμού C. (14)

Microsoft Robotics Developer Studio
Το Microsoft Robotics Developer Studio (Microsoft RDS, MRDS) είναι ένα περιβάλλον βασισμένο σε Windows για τον έλεγχο και προσομοίωση του ρομπότ. Απευθύνεται σε ακαδημαϊκούς και χομπίστες και χειρίζεται μια ευρεία ποικιλία του ρομποτικού υλικού(hardware). Απαιτεί το λειτουργικό σύστημα Microsoft Windows 7. Περιλαμβάνει: ένα εργαλείο οπτικού προγραμματισμού, γλώσσα προγραμματισμού Microsoft Visual για τη δημιουργία και τον εντοπισμό σφαλμάτων των εφαρμογών για το ρομπότ, Web-based και Windows-based διεπαφές, 3D προσομοίωση (συμπεριλαμβανομένης της επιτάχυνσης υλικού(hardware acceleration), εύκολη πρόσβαση σε αισθητήρες και ενεργοποιητές ενός ρομπότ. Η κύρια γλώσσα προγραμματισμού είναι η C#. (15) (16)

RobotC
Είναι μία γλώσσα προγραμματισμού βασισμένη στη γλώσσα C. Η εκτέλεση των προγραμμάτων γίνεται ταχύτατα και τα προγράμματα καταλαμβάνουν μικρό χώρο στη μνήμη. (17)

nxt-python
Το nxt-python είναι ένας οδηγός/διεπαφή για το Lego Mindstorms NXT. Επιτρέπει την επικοινωνία με το NXT μέσω USB και Bluetooth. Γλώσσα προγραμματισμού είναι η python. (18)

4. ΚΑΤΑΣΚΕΥΗ ΤΟΥ ΡΟΜΠΟΤ

Στην ενότητα 4.1, παρουσιάζονται και επεξηγούνται οι λόγοι που διαλέξαμε τη συγκεκριμένη μορφή του ρομπότ και στην συνέχεια, στην ενότητα 4.2, δίνονται αναλυτικά οι οδηγείες, με εικόνες, για την κατασκευή του, βήμα βήμα.

4.1 Σχεδιαστικές αποφάσεις

4.1.1 Ο αισθητήρας ήχου

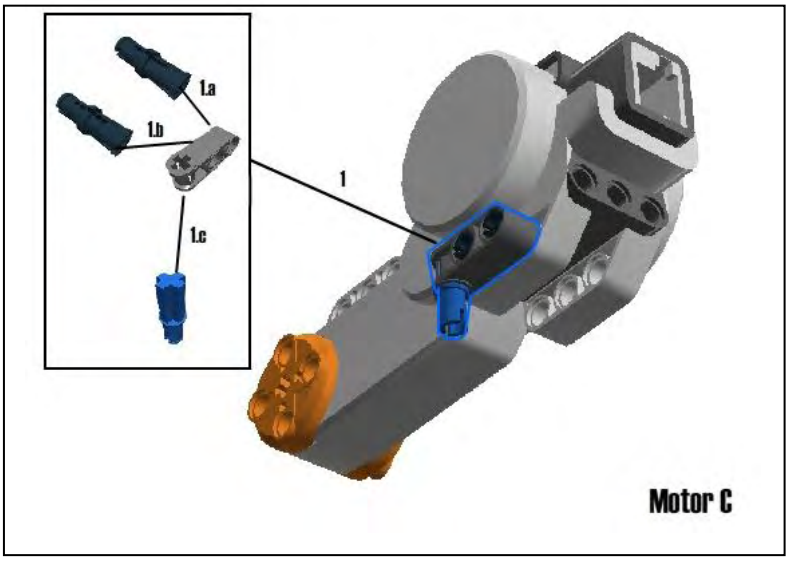
Ο αισθητήρας ήχου είναι αυτός που υποδυκνύει την αρχή και το τέλος στην κίνηση του ρομπότ. Με το χτύπημα των χεριών μας, ξεκινά να ακολουθεί το χαραγμένο μονοπάτι, έως ότου χτυπήσουμε ξανά τα χέρια μας και με αυτόν τον τρόπο στέλνουμε το σήμα για να σταματήσει. Η θέση του αισθητήρα είναι σημαντική γιατί δε θέλουμε να παρεμβάλλονται άλλοι ήχοι (π.χ. από τα σερβομοτέρ), ώστε να τον μπερδέψουν. Για αυτό το λόγο έχει προσαρτηθεί ψηλότερα από το υπόλοιπο ρομπότ.

4.1.2 Ο αισθητήρας απόστασης

Ο αισθητήρας απόστασης πρέπει να ελέγχει για "παραβάτες" κατά την περιπολεία πάνω στο μονοπάτι. Για αυτό το λόγο έχει τοποθετηθεί πάνω σε έναν άξονα ο οποίος περιστρέφεται με τη χρήση του ενός σερβοκινητήρα. Ο σερβοκινητήρας περιστρέφει τον αισθητήρα σε ημικύκλιο, και ελέγχει για εμπόδια από τα αριστερά προς τα δεξιά του ρομπότ. Κατά την κατασκευή του NXT, έπρεπε να ληφθεί υπ’ όψην, να μην υπάρχει άλλο εμπόδιο το οποίο να εντοπίζει ο αισθητήρας, πέρα από τους "εισβολείς" στο πεδίο του.

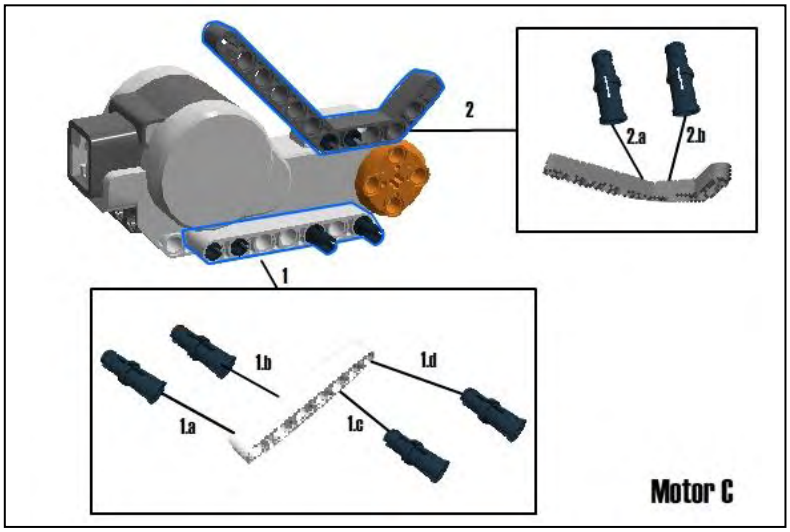
4.2 Η τελική κατασκευή

Εδώ παρουσιάζονται τα βήματα που ακολουθήθηκαν για την κατασκευή του ρομπότ. Το μοντέλο Lego NXT που ακεικονίζεται παρακάτω δημιουργήθηκε με το πρόγραμμα Lego Digital Designer. (19)



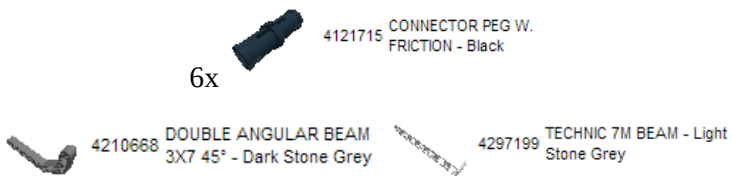
Εικόνα 4.1

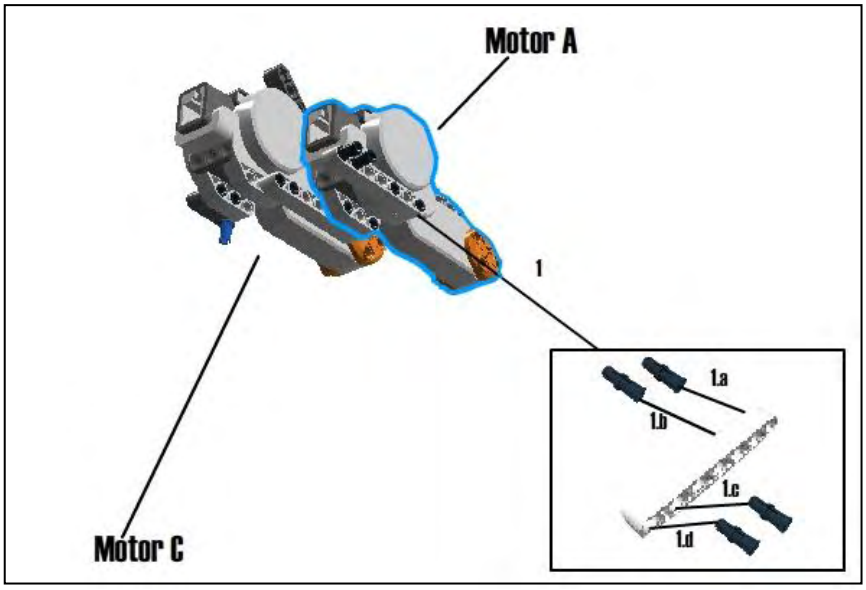
Κομμάτια Lego:



Εικόνα 4.2

Κομμάτια Lego:





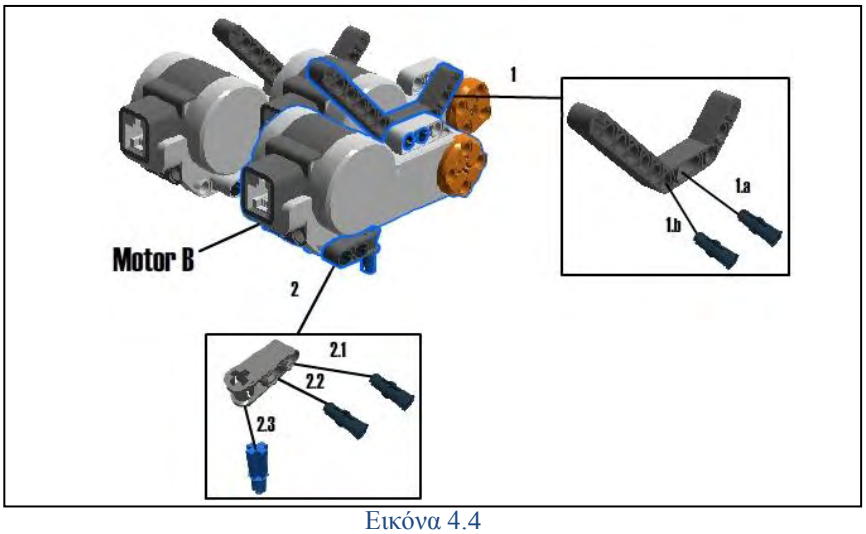
Εικόνα 4.3

Κομμάτια Lego:

4x 4121715 CONNECTOR PEG W. FRICTION - Black

4297008 Tacho Motor - Bright Orange,Dark Stone Grey,Light Stone Grey

4297199 TECHNIC 7M BEAM - Light Stone Grey



Εικόνα 4.4

Κομμάτια Lego:

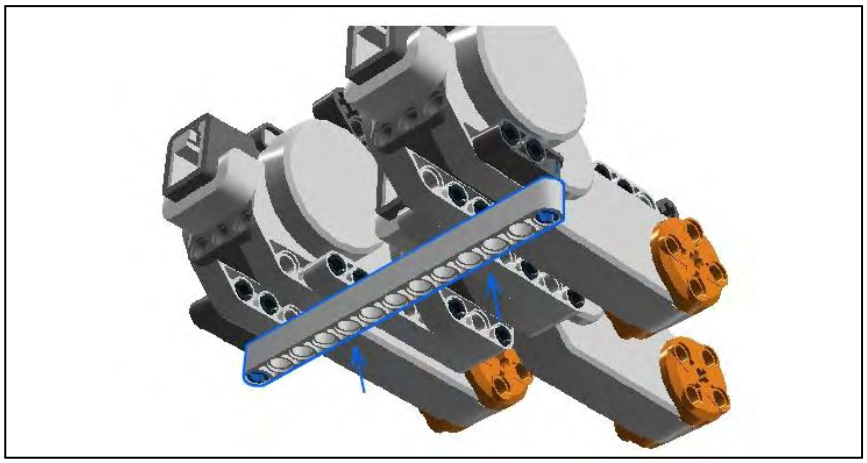
4x 4121715 CONNECTOR PEG W. FRICTION - Black

4210668 DOUBLE ANGULAR BEAM 3X7 45° - Dark Stone Grey

4297199 TECHNIC 7M BEAM - Light Stone Grey

4206482 CONN.BUSH W.FRIC./CROSSALE - Bright Blue

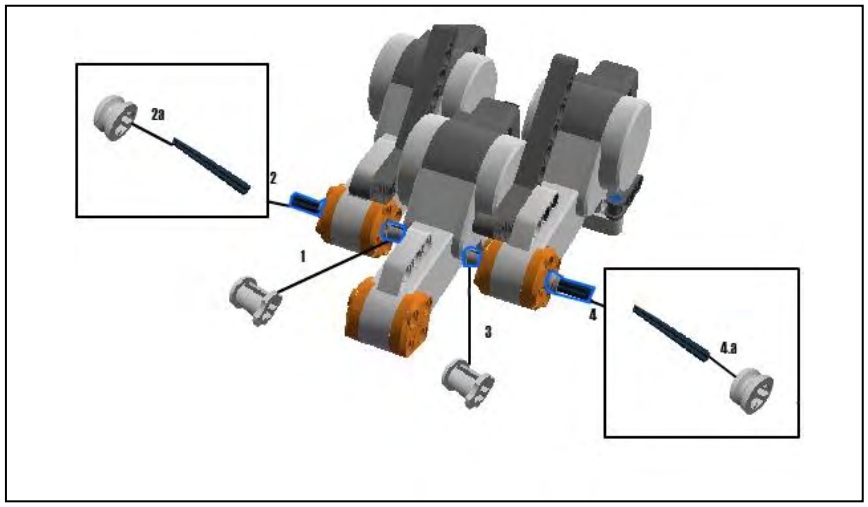
4210857 CROSS BLOCK 3M - Dark Stone Grey



Εικόνα 4.5

Κομμάτια Lego:

4297203 TECHNIC 13M BEAM - Light Stone Grey



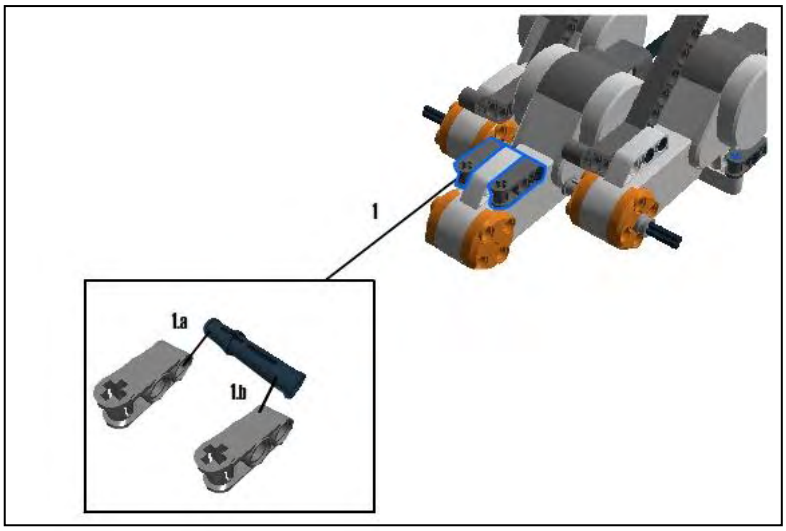
Εικόνα 4.6

Κομμάτια Lego:

370626 CROSS AXLE 6M - Black

4211573 1/2 BUSH - Medium Stone Grey

4211622 BUSH FOR CROSS AXLE - Medium Stone Grey

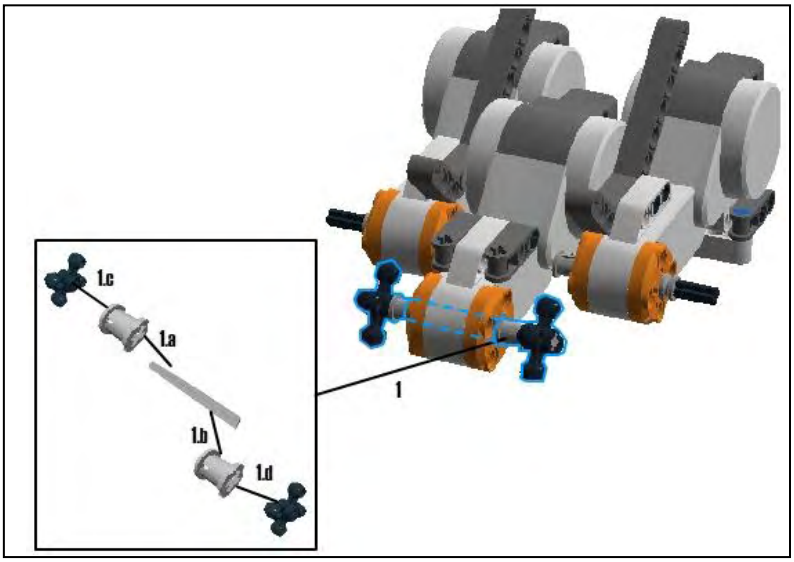


Εικόνα 4.7

Κομμάτια Lego:


2x 4210857 CROSS BLOCK 3M - Dark Stone Grey


655826 CONNECTOR PEG W. FRICTION 3M - Black



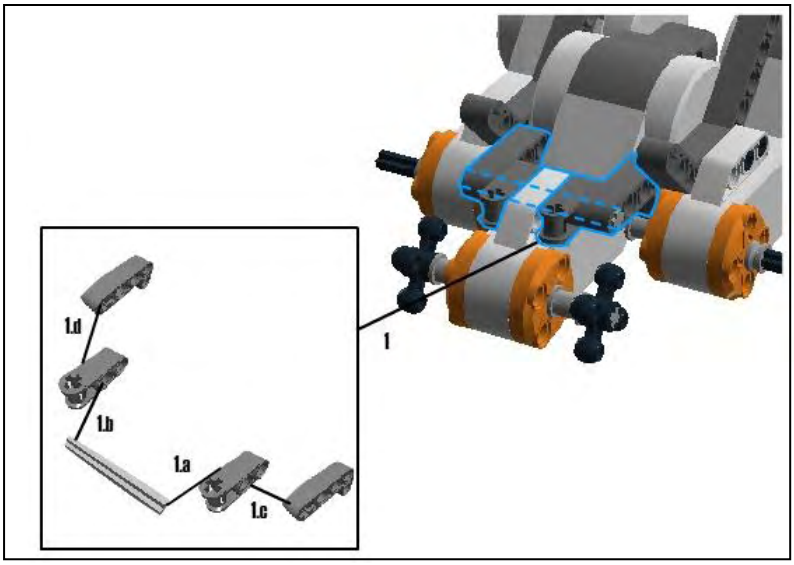
Εικόνα 4.8

Κομμάτια Lego:


2x 4211622 BUSH FOR CROSS AXLE - Medium Stone Grey


2x 4248204 TECHNIC ANGULAR WHEEL - Black


4211805 CROSS AXLE 7M - Medium Stone Grey



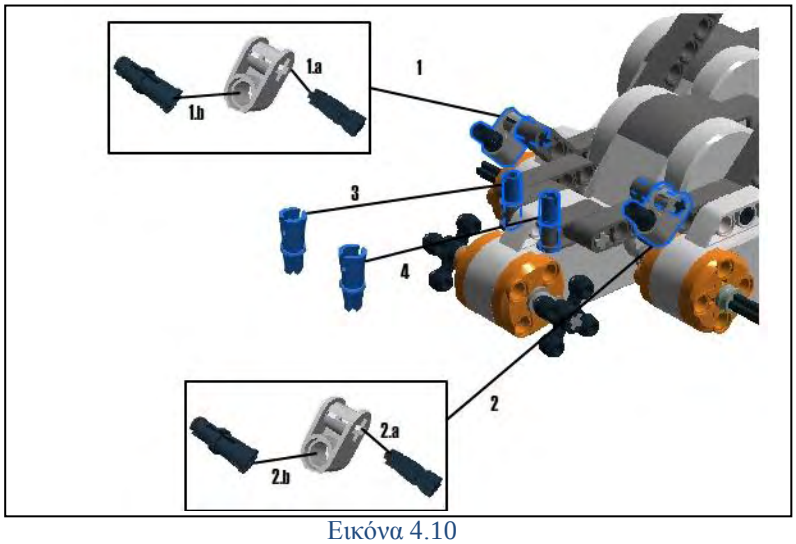
Εικόνα 4.9

Κομμάτια Lego:


2x 4210857 CROSS BLOCK 3M - Dark Stone Grey


2x 4210667 TECHNIC ANG. BEAM 4X2 90 DEG - Dark Stone Grey


4211639 CROSS AXLE 5M - Medium Stone Grey



Εικόνα 4.10

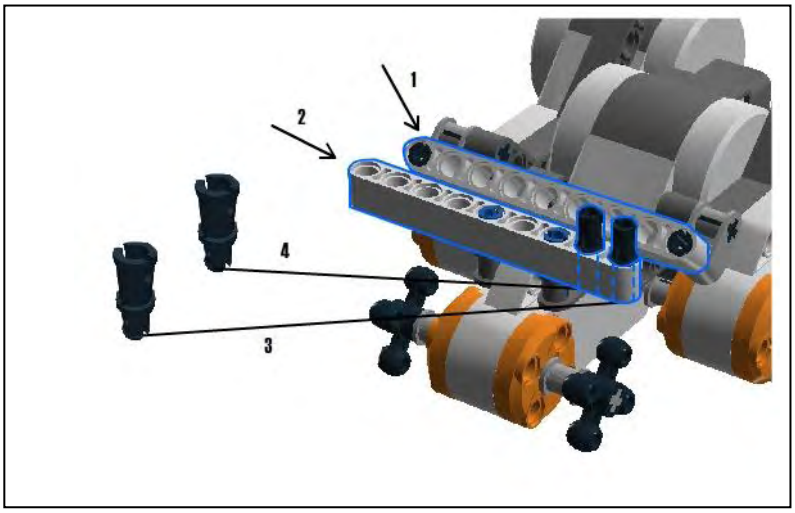
Κομμάτια Lego:


2x 4211775 CROSS BLOCK 90° - Medium Stone Grey


2x 4121715 CONNECTOR PEG W. FRICTION - Black


2x 4206482 CONN.BUSH W.FRIC./CROSSALE - Bright Blue


2x 4109810 2M CROSS AXLE W. GROOVE - Black

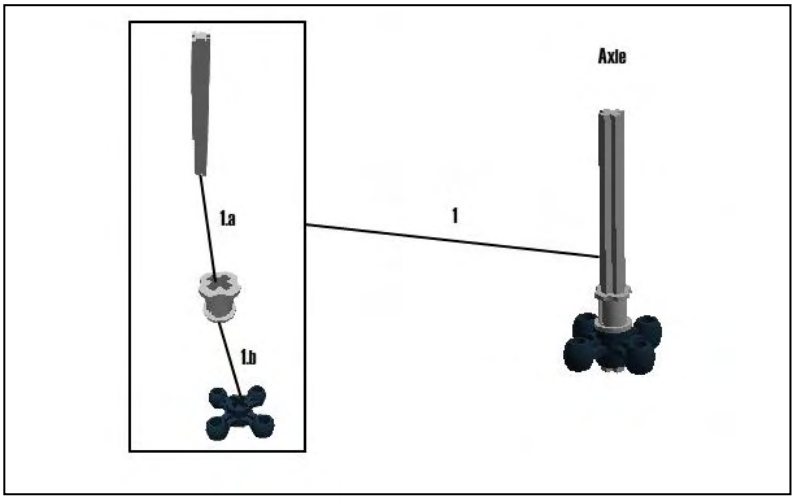


Εικόνα 4.11

Κομμάτια Lego:

2x 4121715 CONNECTOR PEG W. FRICTION - Black

2x 4297202 TECHNIC 9M BEAM - Light Stone Grey



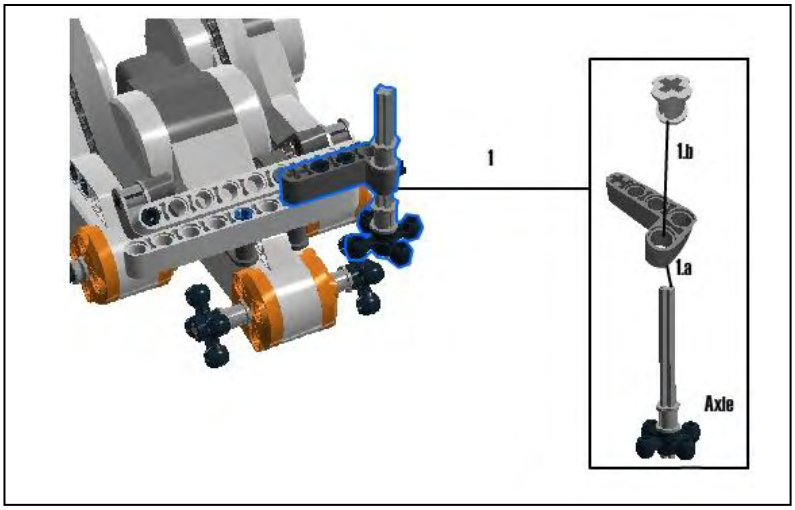
Εικόνα 4.12

Κομμάτια Lego:

4211622 BUSH FOR CROSS AXLE - Medium Stone Grey

4248204 TECHNIC ANGULAR WHEEL - Black

4211805 CROSS AXLE 7M - Medium Stone Grey

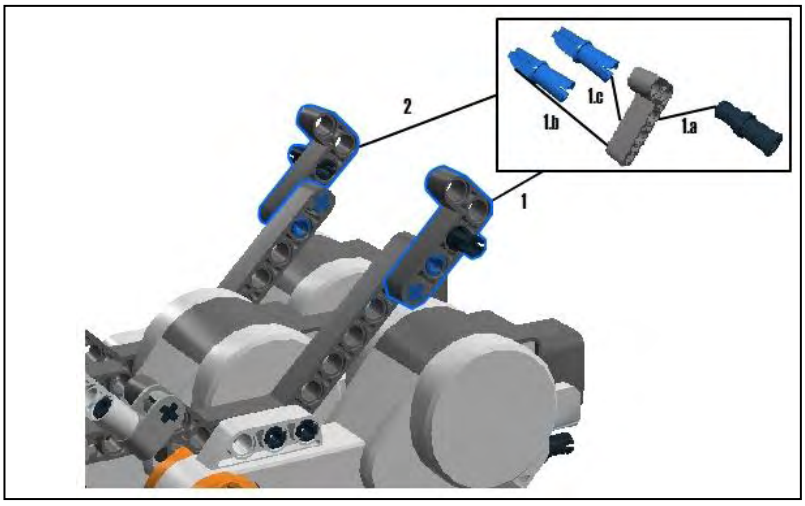


Εικόνα 4.13

Κομμάτια Lego:


4211622 BUSH FOR CROSS AXLE - Medium Stone Grey


4210667 TECHNIC ANG. BEAM 4X2 90 DEG - Dark Stone Grey



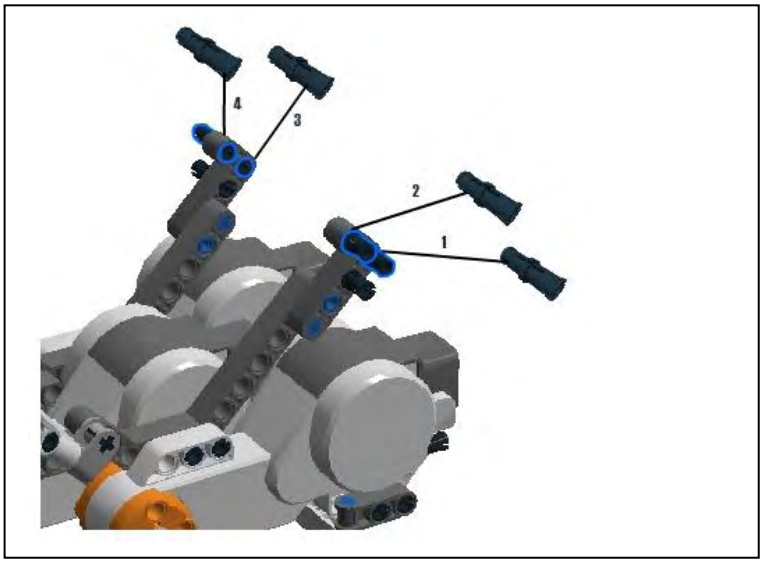
Εικόνα 4.14

Κομμάτια Lego:


2x 4121715 CONNECTOR PEG W. FRICTION - Black


4x 4206482 CONN.BUSH W.FRIC./CROSSALE - Bright Blue


2x 4210667 TECHNIC ANG. BEAM 4X2 90 DEG - Dark Stone Grey

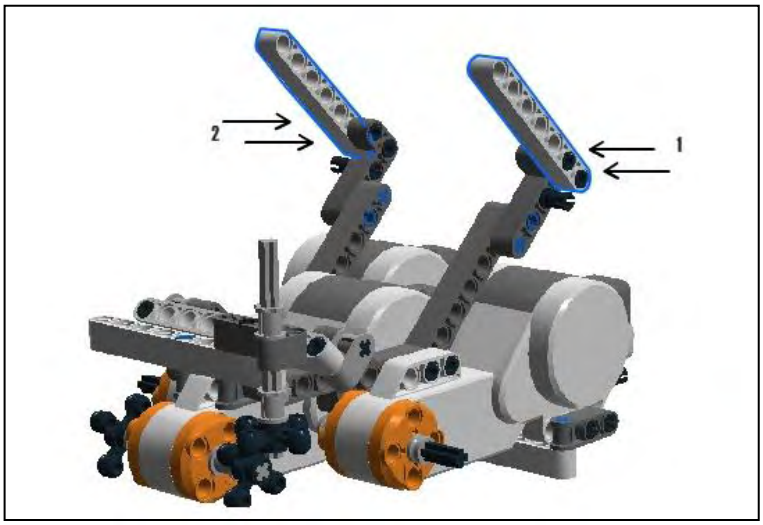


Εικόνα 4.15

Κομμάτια Lego:

4121715 CONNECTOR PEG W. FRICTION - Black

4x

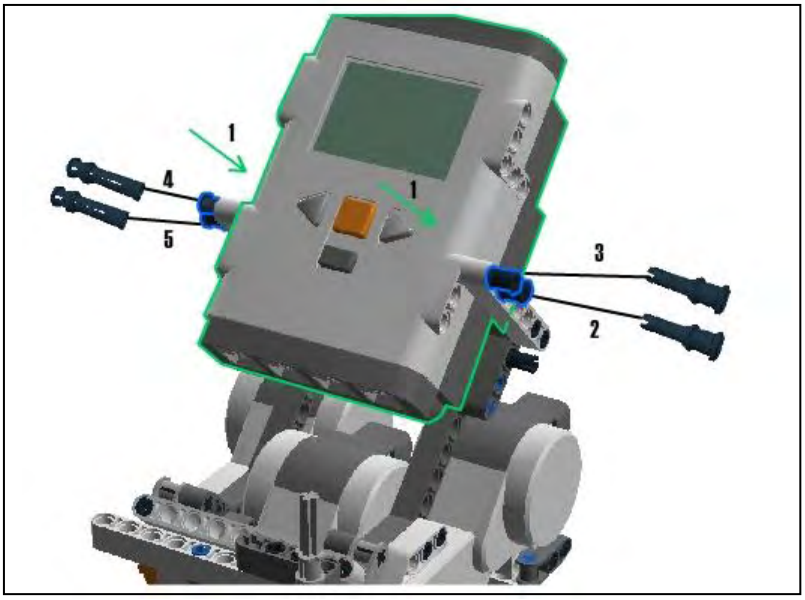


Εικόνα 4.16

Κομμάτια Lego:

4297199 TECHNIC 7M BEAM - Light Stone Grey

2x

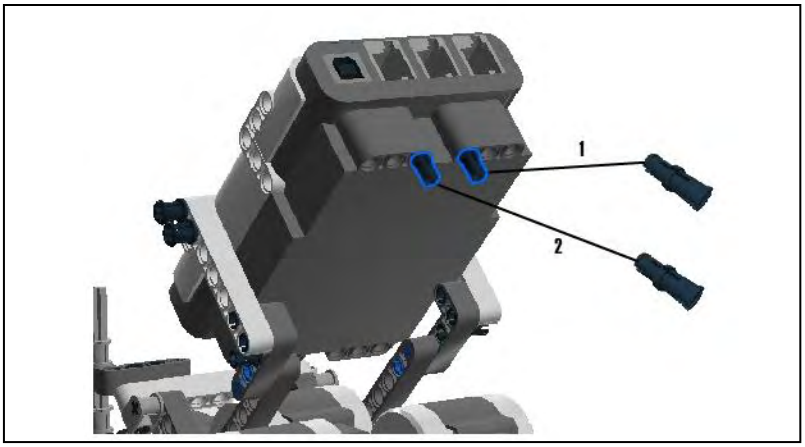


Εικόνα 4.17

Κομμάτια Lego:

4x  4107742 2M FRIC. SNAP W/CROSS HOLE - Black  6034375 NXT - Black,Bright Orange,Sand Green,Medium Stone Grey,Dark Stone Grey,Light Stone Grey,Cool silver, drum lacq

---



Εικόνα 4.18

Κομμάτια Lego:

2x  4121715 CONNECTOR PEG W. FRICTION - Black

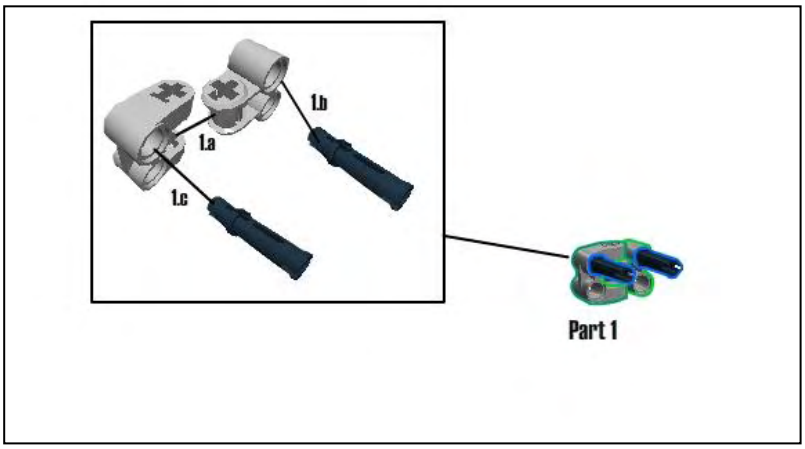
Εικόνα 4.19

Κομμάτια Lego:


2x 4297209 TYRE BALOON WIDE Ø56 X 26 - Black


2x 4297210 RIM WIDE W.CROSS 30x20 - Medium Stone Grey

Ενώνουμε ένα λάστιχο και μία ζάντα και μας δίνουν τη ρόδα που εφαρμόζουμε πάνω στο ρομπότ.



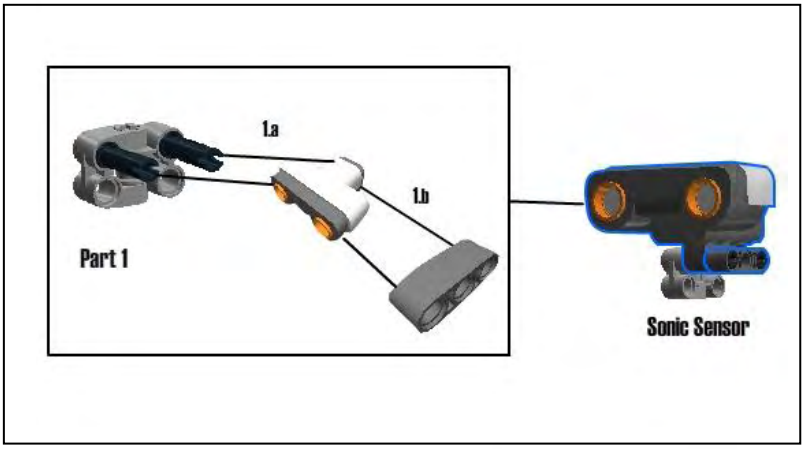
Εικόνα 4.20

Κομμάτια Lego:


4211629 TECHNIC CROSS BLOCK 2X1 - Medium Stone Grey


4211889 TECHNIC CROSS BLOCK/FORK 2X2 - Medium Stone Grey


2x 655826 CONNECTOR PEG W. FRICTION 3M - Black



Εικόνα 4.21

Κομμάτια Lego:

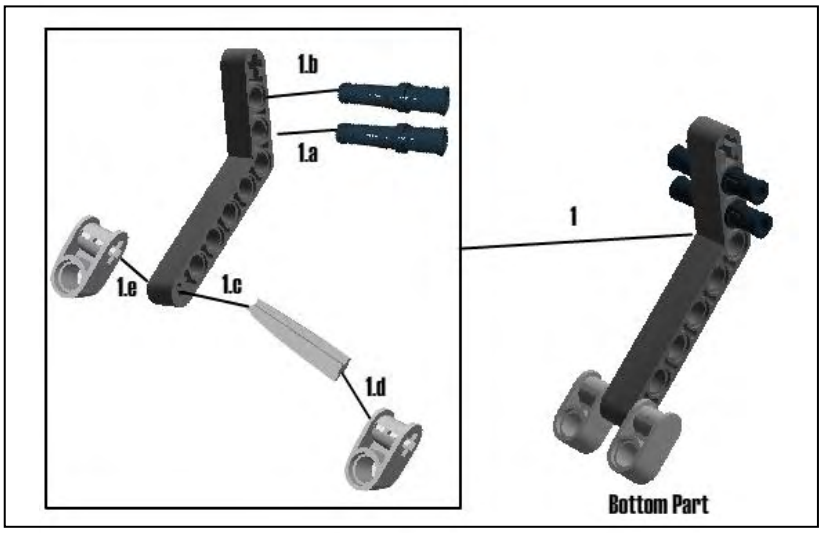

4297174 Ultrasound sensor - Bright Orange,Medium Stone Grey,Dark Stone Grey,Light Stone Grey


4210751 TECHNIC 3M BEAM - Dark Stone Grey

Εικόνα 4.22

Κομμάτια Lego:

Ο αισθητήρας που συναρμολογήσαμε στην εικόνα 4.21



Εικόνα 4.23

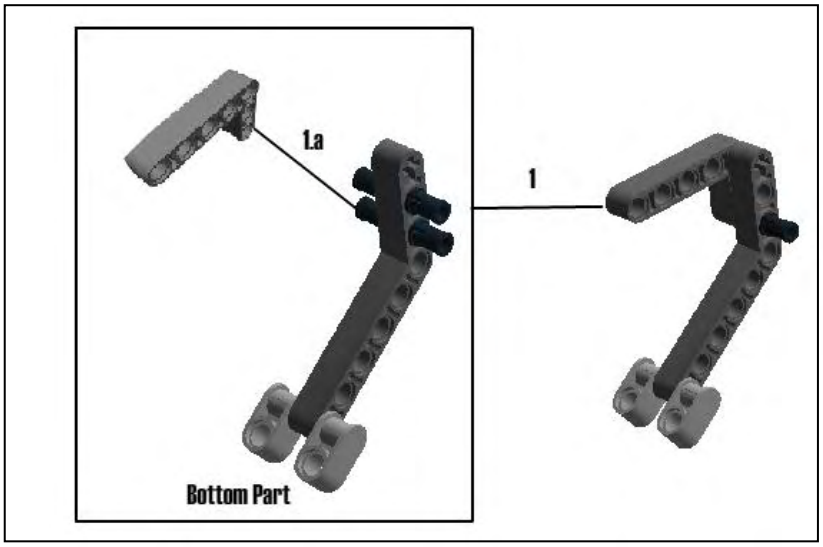
Κομμάτια Lego:


4210638 TECHNIC ANGULAR BEAM 4X6 - Dark Stone Grey


4211815 CROSS AXLE 3M - Medium Stone Grey


2x 655826 CONNECTOR PEG W. FRICTION 3M - Black

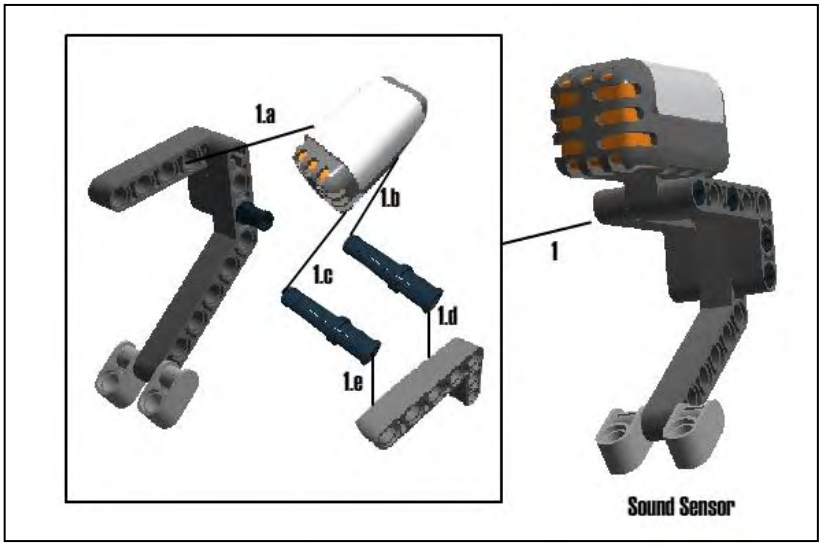

2x 4211775 CROSS BLOCK 90° - Medium Stone Grey



Εικόνα 4.24

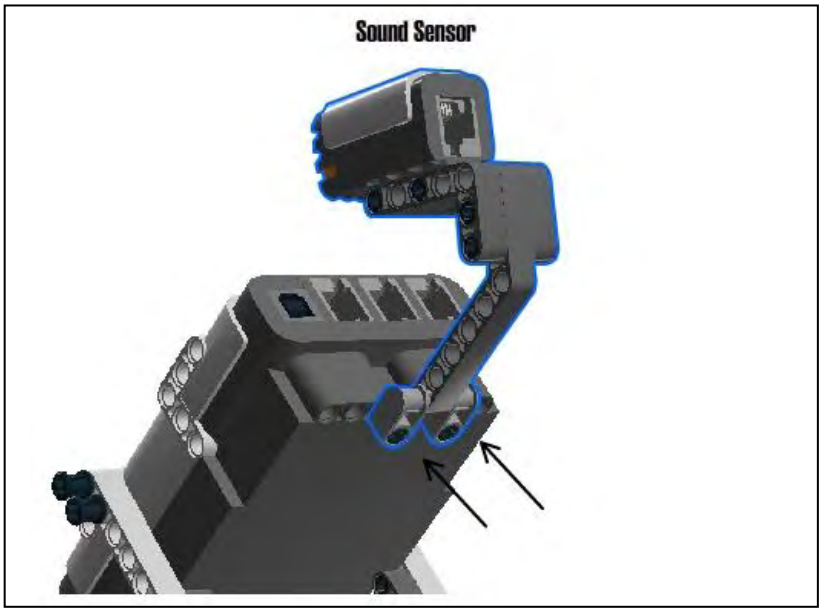
Κομμάτια Lego:


4210667 TECHNIC ANG. BEAM 4X2 90 DEG - Dark Stone Grey



Εικόνα 4.25

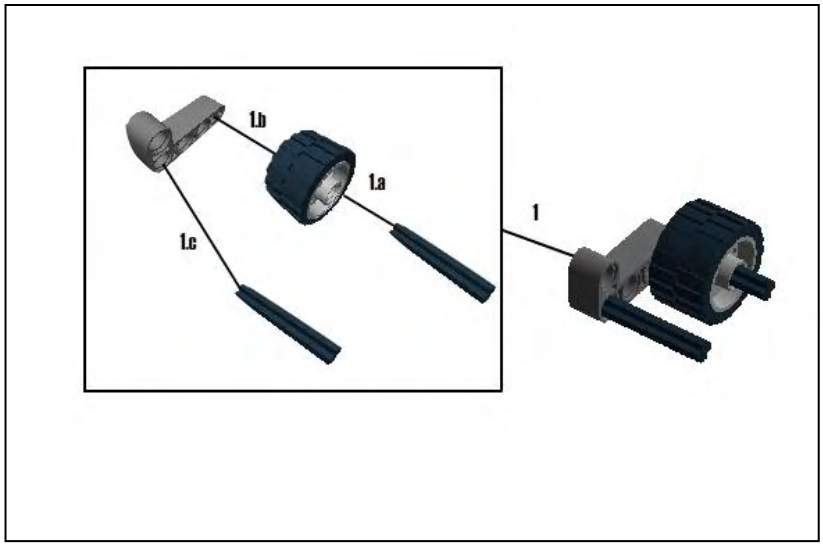
Κομμάτια Lego:



Εικόνα 4.26

Κομμάτια Lego:

Ο αισθητήρας που συναρμολογήσαμε στην εικόνα 4.25



Εικόνα 4.27

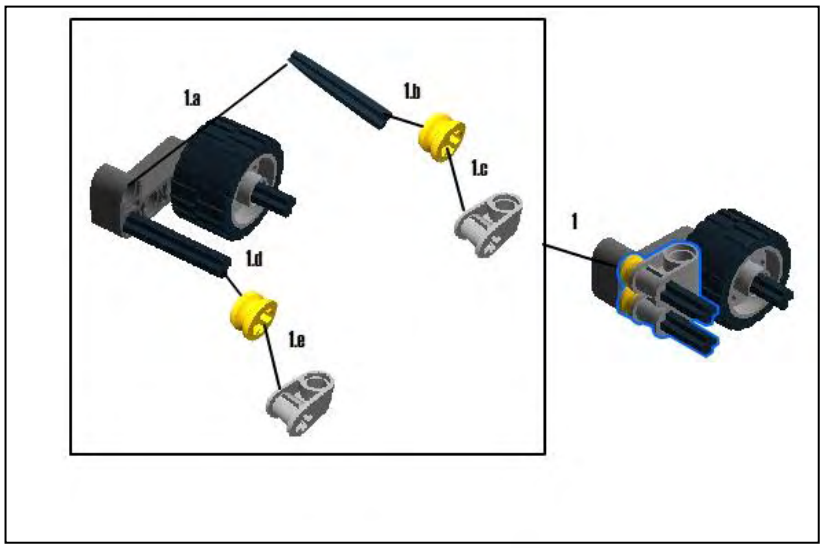
Κομμάτια Lego:

2x  370526 CROSS AXLE 4M - Black

 4299119 RIM WIDE 18x14 W. HOLE Ø4.8 - Medium Stone Grey

 4153005 TYRE LOW WIDE Ø24 X 14 - Black

Ενώνουμε τη ζάντα και το λάστιχο για να μας δώσουν τη ρόδα στήριξης

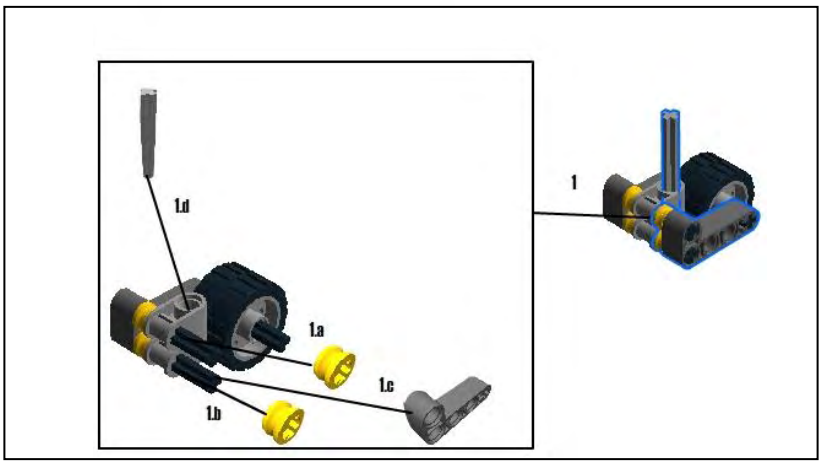


Εικόνα 4.28

Κομμάτια Lego:

2x  4239601 1/2 BUSH - Bright Yellow

2x  4211775 CROSS BLOCK 90° - Medium Stone Grey



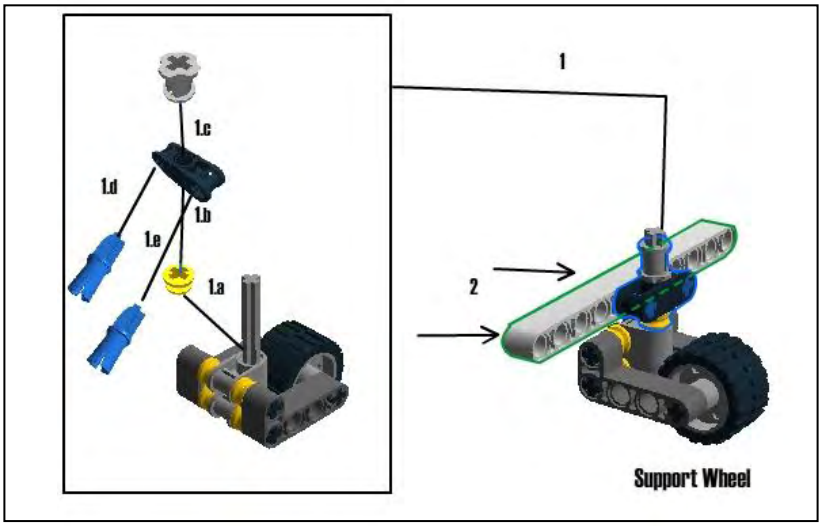
Εικόνα 4.29

Κομμάτια Lego:


4211639 CROSS AXLE 5M - Medium Stone Grey


4210667 TECHNIC ANG. BEAM 4X2 90 DEG - Dark Stone Grey


2x 4239601 1/2 BUSH - Bright Yellow



Εικόνα 4.30

Κομμάτια Lego:


4239601 1/2 BUSH - Bright Yellow


4121667 DOUBLE CROSS BLOCK - Black


2x 4206482 CONN.BUSH W.FRIC./CROSSALE - Bright Blue


4211622 BUSH FOR CROSS AXLE - Medium Stone Grey


4297200 TECHNIC 11M BEAM - Light Stone Grey

Εικόνα 4.31

Κομμάτια Lego:

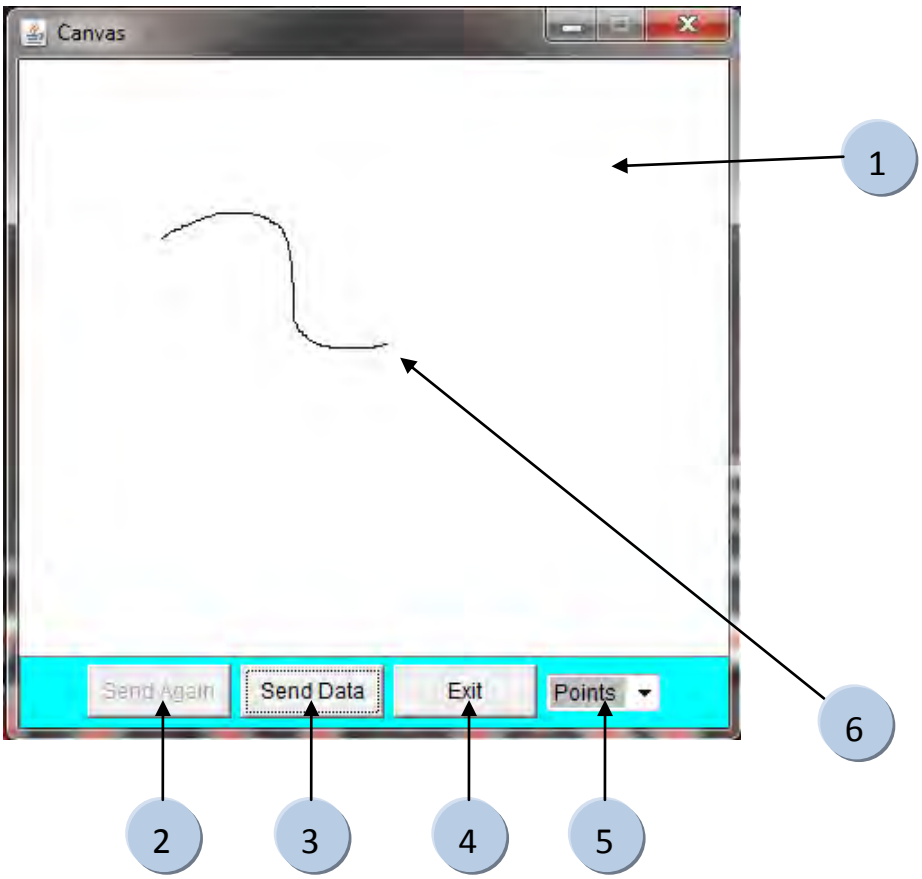
Προσθέτουμε τη ρόδα στήριξης που συναρμολογήσαμε στην εικόνα 4.30, στο υπόλοιπο ρομπότ και φτάνουμε στο τέλος της κατασκευής μας.

5. ΓΡΑΦΙΚΟ ΠΕΡΙΒΑΛΛΟΝ ΧΡΗΣΤΗ (GUI)

Σε αυτό το κεφάλαιο θα περιγραφεί και θα αναλυθεί το γραφικό περιβάλλον στο οποίο ο χρήστης καλείται να χαράξει το σχήμα το οποίο στη συνέχεια θα ακολουθήσει το ρομπότ.

5.1 Τα μέρη που αποτελούν το γραφικό περιβάλλον (GUI)

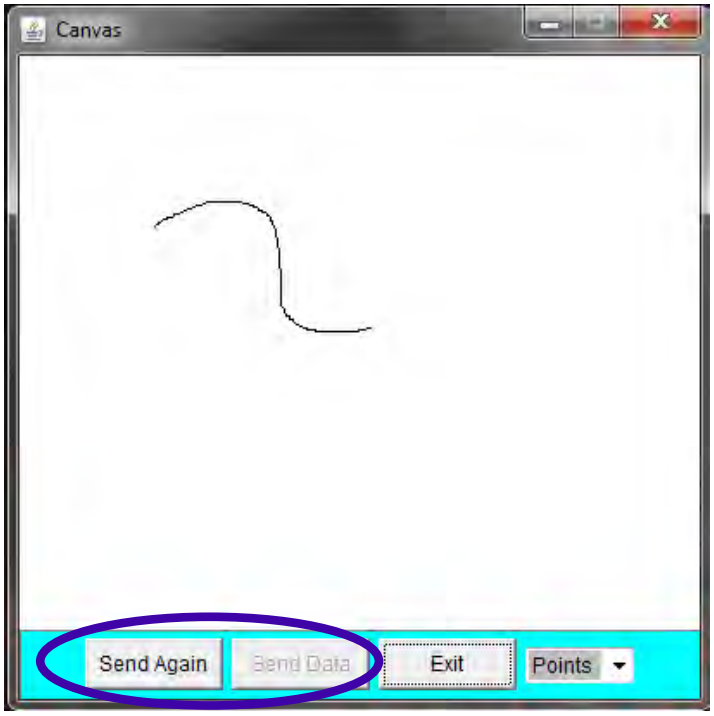
Στην παρακάτω εικόνα παρουσιάζονται το γραφικό περιβάλλον με ένα μονοπάτι το οποίο χαράξαμε.



Παρακάτω αναλύονται τα συστατικά του μέρη:

1. Το πλαίσιο στο οποίο ο χρήστης χαράζει το μονοπάτι της επιλογής του. Οι διαστάσεις του είναι 400*400 και έχει προεπιλεχθεί να μην επιτρέπεται η αλλαγή μεγέθους του.

2. Το πλήκτρο Send Again μας δίνει τη δυνατότητα να στείλουμε ξανά το μονοπάτι με την προϋπόθεση να το έχουμε στείλει στο ρομπότ μία φορά. Είναι αρχικά απενεργοποιημένο.
3. Με το πλήκτρο Send Data στέλνουμε στο ρομπότ το μονοπάτι που χαράξαμε. Μόλις αυτό σταλεί και αποσυνδέσουμε το κάλωδιο USB από το NXT, απενεργοποιείται το Send Data και ενεργοποιείται το Send Again. Στην παρακάτω εικόνα φαίνεται η εναλλαγή ενεργοποιημένου-απανεργοποιημένου στα πλήκτρα:



4. Το πλήκτρο Exit, κλείνει το τρέχον παράθυρο.
5. Σε αυτό το drop-down μενού επιλογών, μας δίνεται η δυνατότητα να επιλέξουμε ο τρόπος που θα σχηματίζουμε το μονοπάτι. Προς το παρών, έχουμε μόνο μία επιλογή: σημείο προς σημείο, δηλαδή όπως θα ζωγραφίζαμε μία γραμμή στο χαρτί.
6. Το μονοπάτι που μόλις χαράξαμε.

5.2 Αναλυτική λειτουργία των πλήκτρων

Send Data

Πατώντας το, εκτελείται το παρακάτω κομμάτι κώδικα:

```
Send.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        Path.PrintMap("c:\\map.txt");
        Path.BuildTable();
        SendAgain.setEnabled(true);
        Send.setEnabled(false);
    }

});
```

Με απλά λόγια, καλέι τις μεθόδους PrintMap και BuildTable από την κλάση Path, που εκτυπώνουν το μονοπάτι σε ένα αρχείο με επέκταση .txt και το αποθηκεύουν, αντίστοιχα. Στη συνέχεια, ενεργοποιεί το πλήκτρο Send Again ενώ απενεργοποιείται το ίδιο.

Send Again

Πατώντας το, εκτελείται το παρακάτω κομμάτι κώδικα:

```
SendAgain.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        SimplifiedPath.CalculatePath();
    }

});
```

Εδώ καλείται η μέθοδος CalculatePath της κλάσης SimplifiedPath, η οποία ξαναστέλνει το υπολογισμένο μονοπάτι.

Exit

Με το πλήκτρο Exit καλούμε τη μέθοδο exit της System για να κλείσουμε το πρόγραμμά μας.

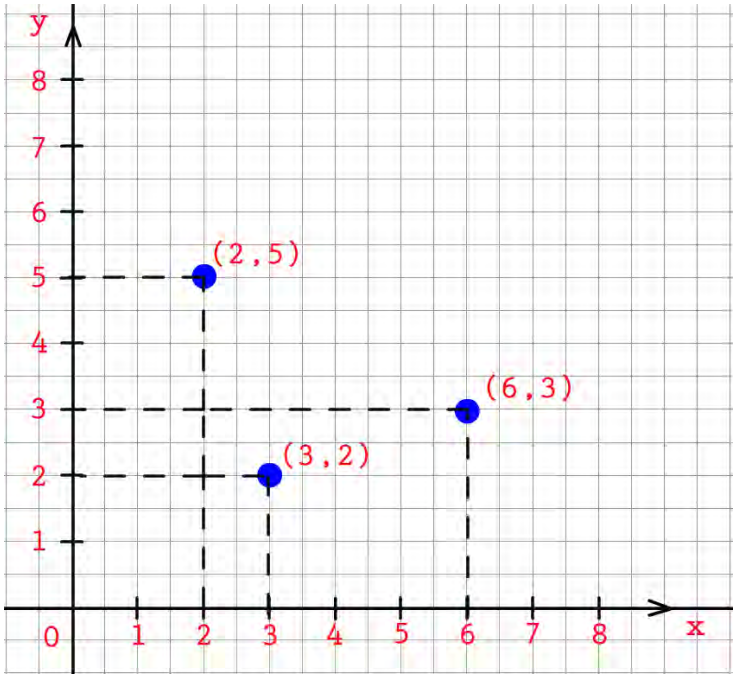
```
Exit.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        System.exit(0);
    }

});
```

6. ΤΟ ΜΟΝΟΠΑΤΙ

Σε αυτό το κεφάλαιο παρουσιάζεται αναλυτικά ο τρόπος που διαβαζουμε το μονοπατι και το μεταφράζουμε σε δεδομένα που μας είναι απαραίτητα για να το ακολουθήσει το ρομπότ μας.

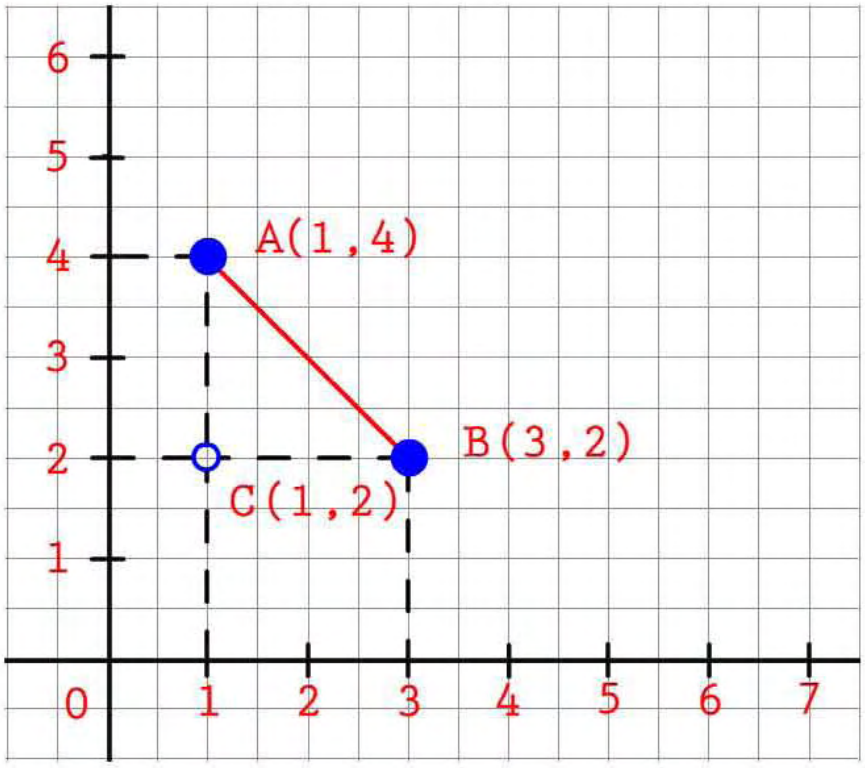
6.1 Τι είναι σημείο στο επίπεδο

Αν εστιάσουμε πάνω στη γραμμή του μονοπατιού μας, θα παρατηρήσουμε ότι αποτελείται από πολλές μικροσκοπικές κουκκίδες, τα pixels. Κάθε pixel το θεωρούμε ως σημείο. Κάθε σημείο στο διδιάστατο επίπεδο, σύμφωνα με την Ευκλείδια γεωμετρία, αντιπροσωπεύεται από ένα ζεύγος αριθμών (x,y) που αντιστοιχούν στον οριζόντιο άξονα x και στον κάθετο άξονα y.



6.2 Υπολογισμός ευθύγραμμου τμήματος

Μία ευθεία γραμμή στο επίπεδο, αντιπροσωπεύει ένα αντίκείμενο άπειρου μήκους αλλά αμελητέου βάθους και πλάτους. Ένα ευθύγραμμο τμήμα, είναι μία ευθεία γραμμή της οποίας το μήκος καθορίζεται από δύο σημεία στο επίπεδο. Έστω το παρακάτω σχήμα:



Έχουμε τα σημεία Α(1,4) και Β(3,2). Το ερώτημα είναι πώς θα βρούμε το μήκος του ευθύγραμμου τμήματος ανάμεσα στα δύο σημεία. Για να το βρούμε, θεωρούμε ένα ακόμα σημείο για να βοηθηθούμε, το C(1,2) το οποίο βρίσκεται στο σημείο με x ίσο με το x του Α και y ίσο με το y του Β. Θα ήταν το ίδιο αν θεωρούσαμε για C το σημείο με x ίσο με το x του Β και y ίσο με το y του Α. Παρατηρούμε πως σχηματίζεται ένα ορθογώνιο τρίγωνο, στο οποίο έχουμε γνωστές τις δύο πλευρές του και ψάχνουμε την τελευταία. Στα ορθογώνια τρίγωνα η μεγαλύτερη πλευρά, η υποτείνουσα, βρίσκεται με τον τύπο:

$AB^2 = BC^2 + AC^2 \Leftrightarrow AB = \sqrt{BC^2 + AC^2} \Leftrightarrow AB = \sqrt{(3-1)^2 + (4-2)^2} \Leftrightarrow AB = \sqrt{8} \Leftrightarrow$

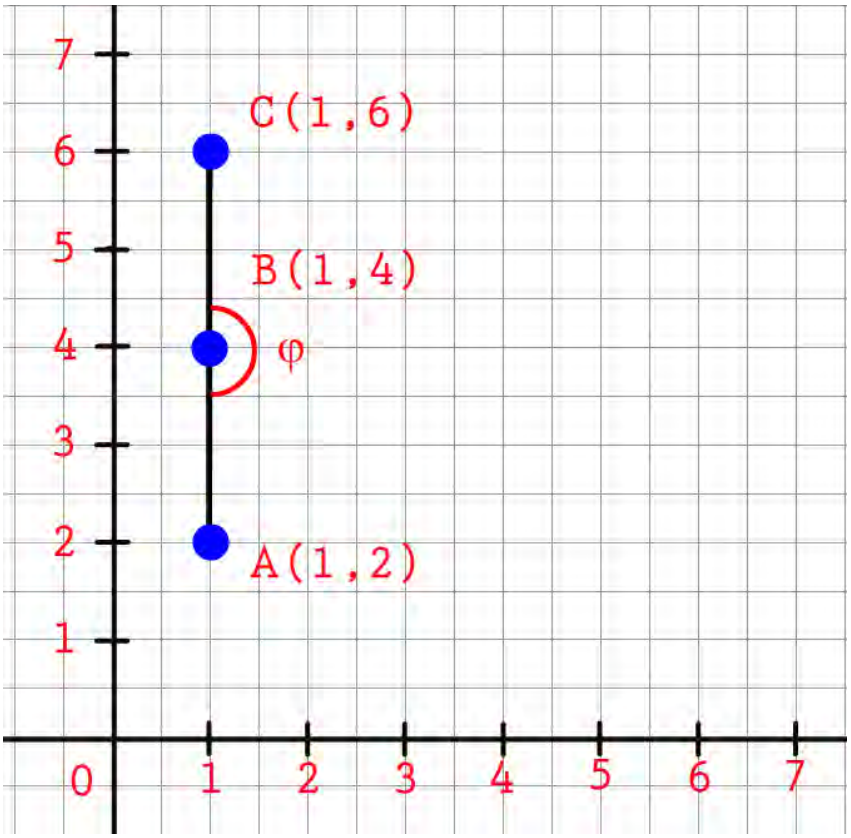
$\Leftrightarrow AB = 2 * \sqrt{2}$

Επομένως γενικά έχουμε αν $A=(x_A, y_B)$, $B=(x_B, y_B)$ και $C=(x_C, y_C)$:

$AB^2 = BC^2 + AC^2 \Leftrightarrow AB = \sqrt{BC^2 + AC^2} \Leftrightarrow AB = \sqrt{(x_B - x_C)^2 + (y_A - y_C)^2}$

6.3 ***Υπολογισμός γωνίας μεταξύ δύο ευθύγραμμων τμημάτων***

Το επόμενο βήμα είναι να υπολογίσουμε τη γωνία που σχηματίζεται μεταξύ δύο διαδοχικών ευθύγραμμων τμημάτων. Ένα πιθανό πρόβλημα που θα αντιμετωπίσουμε είναι το παρακάτω:

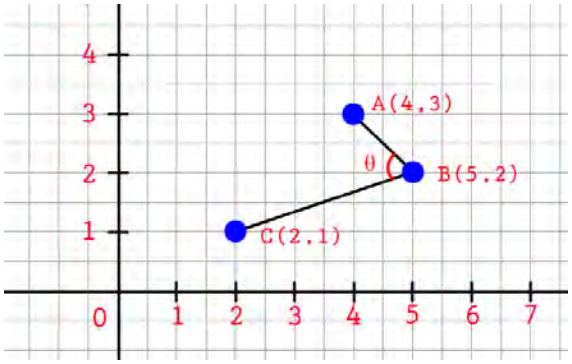


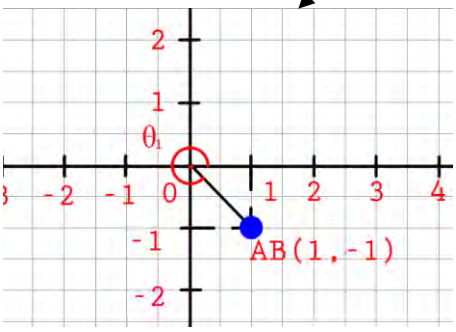
Για να βρούμε τη γωνία φ έχουμε:

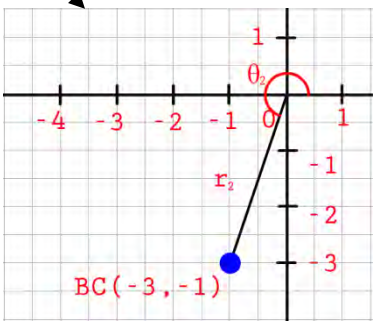
Ο λόγος της ΑΒ: $\lambda_{AB} = \dfrac{(y_B - y_A)}{(x_B - x_A)}$

Παρατηρούμε ότι $(x_B - x_A) = 0$, επομένως η διαίρεση είναι αδύνατη.

Για να ξεπεράσουμε αυτό το εμπόδιο, μπορούμε να μετατρέψουμε τις συντεταγμένες από το ορθοκανονικό σύστημα στο σύστημα πολικών συντεταγμένων. Έστω A=(4,3), B=(5,2), C=(2,1). Από τα σημεία έχω δύο ευθείες που ξεκινούν από την αρχή των αξόνων, τις AB=$((x_B - x_A),(y_B - y_A))$ = (1,1) και BC=$((x_C - x_B),(y_C - y_B))$ =(-3,-1) :







Όπου $\theta_1 = \arctan\frac{AB_y}{AB_x}$ = arctan(-1) = 5,5 rad και $\theta_2 = \arctan\frac{BC_y}{BC_x}$ = arctan3 = 4,39 rad

Στη συνέχεια μετατρέπουμε την κάθε γωνία από ακτίνια σε μοίρες: $\theta_1 = \frac{180}{3.14} * 5,5 =$ 315^o και $\theta_2 = \frac{180}{3.14} * 4,39 = 251^o$

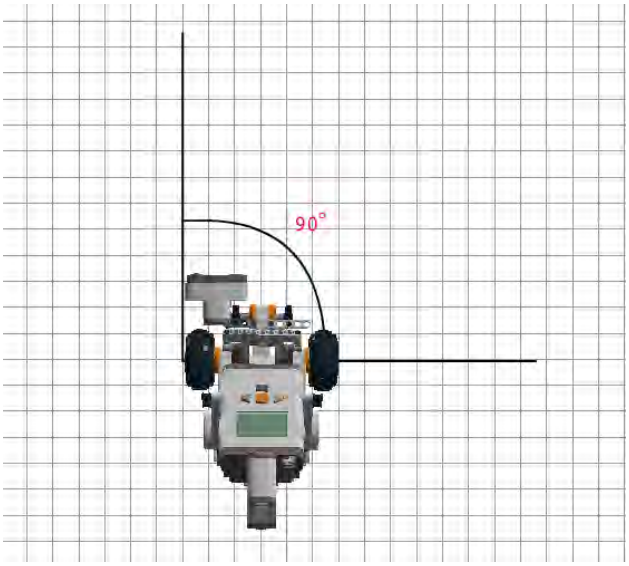
Τέλος, για να βρούμε τη γωνία μεταξύ των ευθειών AB και BC αρκεί να κάνουμε την πράξη $\theta_1 - \theta_2 = 315-251 = 64^o$

6.4 Υπολογισμός της γωνίας σε περιστροφές ρόδας

Έχοντας βρει τη γωνία που θέλουμε στρίψει το ρομπότ, σαν επόμενο βήμα έχουμε τον υπολογισμό των περιστροφών που θα εκτελέσει η ρόδα ώστε το ρομπότ να περιστραφει τόσες μοίρες.

Αρχικά υπολογίζουμε την περίμετρο της ρόδας: c = π*d, όπου c η περίμετρος, d=56 mm η διάμετρος και π=3,14. Άρα έχουμε **c=176mm** περίπου. Επίσης θα μας χρειαστεί και η απόσταση μεταξύ των ροδών που είναι **k=125 mm**.

Στο παρακάτω παράδειγμα θα βρούμε πόσες περιστροφές χρειάζεται να κάνει η ρόδα για να στρίψει το ρομπότ 90^o.



Για να κάνει στροφη 360 μοιρών, πρέπει να διαγράψει κύκλο με διάμετρο την απόσταση των ροδών του. Επομένως η περίμετρος κύκλου είναι k*3,14 = 125*3.14 = 392,5 mm. Συνεπώς για να υπολογίσουμε τις μοίρες μεριστροφής της ρόδας αρκεί να υπολογίσουμε $\frac{392.5}{176}$ =2,2 περιστροφές ή 2,2 * 360 = 792 μοίρες. Άρα για 90^o έχουμε $\frac{792}{4}$ = 198 μοίρες περιστροφής της ρόδας.

Συμπερασματικά ισχύει:

$$\frac{\text{(Γωνία που θελουμε να στρίψει το ρομπότ * 2* π * Απόσταση μεταξύ των ροδών)}}{\text{Περίμετρος ρόδας}}$$

7. ΚΙΝΗΣΗ ΤΟΥ ΡΟΜΠΟΤ

Σε αυτό το κεφάλαιο, αναλύονται τα πάντα γύρω από την κίνηση του ρομπότ, αρχίζοντας από την αποστολή δεδομένων από τον υπολογιστή μέχρι το πώς θα καταλάβει το NXT πότε και που να κινηθεί.

7.1 ΛήψηΔεδομένων

Πριν αρχίσει να ακολουθεί το μονοπάτι, το ρομπότ μας πρέπει να έχει τροφοδοτηθεί με τα δεδομένα από τον υπολογιστή. Η διαδικασία με την οποία μπορεί να γίνει αυτό είναι απλή και έχει ως εξής:

1. Εκκινούμε το πρόγραμμά μας από μεριάς του ρομπότ
2. Εκκινούμε το πρόγραμμά μας από μεριάς του υπολογιστή
3. Συνδέουμε το καλώδιο USB στον υπολογιστή και στο ρομπότ
4. Χαράζουμε το μονοπάτι στο γραφικό μας περιβάλλον
5. Πατάμε το πλήκτρο Send Data στο γραφικό περιβάλλον
6. Τα δεδομένα αποστέλονται στο ρομπότ και όταν φτάσουν το ηχείο του NXT θα ηχήσει 3 φορές.
7. Αποσυνδέουμε το καλώδιο
8. Αφήνουμε το ρομπότ στο πάτωμα
9. Κανουμε ένα δυνατό ήχο και το ρομπότ θα ξεκινήσει

Το ρομπότ μας θα σταματήσει μόλις κάνουμε ξανά ένα δυνατό ήχο.

7.2 Ξεκινώντας τα δύο νήματα(threads)

Αφού λάβει τα δεδομένα και ακούσει τον ήχο που κάνουμε, το ρομπότ έχει πάρει ότι δεδομένα χρειάζεται. Αρχικοποιεί την κλάση DataExchange που δημιουργήσαμε για την ανταλλαγή δεδομένων ανάμεσα στα νήματα και ξεκινάει τα δύο νήματα: ένα νήμα τύπου IntruderCheck για τον έλεγχο για εμπόδια και ένα νήμα τύπου PathFollower για την κίνηση των τροχών πάνω στο μονοπάτι.

```
DataExchange DE;
IntruderCheck Check;
PathFollower PathFollow;

DE = new DataExchange();
DE.setCMD(0);
LCD.clear();
Check = new IntruderCheck(DE);
new Thread(Check).start();
PathFollow = new PathFollower(DE, coords.length, segments.length, angles.length, coords, segments, angles);
new Thread(PathFollow).start();
```

7.3 *Έλεγχος για εμπόδια*

Καθ' όλη τη διαδρομή ο αισθητήρας απόστασης κοιτάζει ευθεία και στέλνει σήμα περιμένοντας την ανάκλασή του. Όταν εντοπίσει ένα αντικείμενο σε μία μικρή απόσταση τότε θέτει στην DataExchange τη μεταβλητή CMD ίση με ένα.

```
uss.ping();
if (uss.getDistance()<20)
{
    DEInfo.setCMD(1);
}
```

7.4 *Ακολουθώντας το μονοπάτι*

Το ρομπότ για να ακολουθήσει το μονοπάτι, χρειάζεται τον πίνακα move με τα μήκη των ευθύγραμμων τμημάτων που έχει να ακολουθήσει και τον πίνακα turn που περιέχει τις γωνίες που πρέπει να στρίψει. Αρχίζει να ακολουθεί το μονοπάτι μόλις ακούσει ένα δυνατό ήχο από εμάς. Όσο δεν έχει εντοπίσει εμπόδια, συνεχίζει να ακολουθεί κανονικά το μονοπάτι. `if(DEInfo.getCMD()==0)`

Μόλις εντοπίσει κάποιο, η CMD γίνεται ίση με ένα και τότε καλείται η μέθοδος overtake που βρίσκεται μέσα στην κλάση PathFollower.

Αλλιώς συνεχίζει, να ακολουθεί το μονοπάτι μπρος πίσω

```
if (Itinerary==0)
{
    Motor.B.rotate(move[i], true);
    Motor.C.rotate(move[i], true);
    LCD.drawInt(move[i],0,0);

    if (i>turn.length-1)
    {
        i=moveLength-1;
        Itinerary=1;
        threeSixty();
        LCD.clear();
        Sound.beep();
    }
    else
    {
        if (turn[i]>0)
        {
            Motor.B.rotate(turn[i]);
        }
        else if(turn[i]<0)
        {
            Motor.C.rotate(Math.abs(turn[i]));
        }
        LCD.drawInt(turn[i],5,0);
        i++;
    }
```

```
}
else
{
    Motor.B.rotate(move[i], true);
    Motor.C.rotate(move[i], true);
    LCD.drawInt(move[i],0,1);
    i--;
    if(i<0)
    {
        i=0;
        Itinerary=0;
        threeSixty();
        LCD.clear();
        Sound.beep();
    }
    else
    {
        if (turn[i]>0)
        {
            Motor.C.rotate(turn[i]);
        }
        else if(turn[i]<0)
        {
            Motor.B.rotate(Math.abs(turn[i]));
        }
    }

}
```

8. ΕΜΠΟΔΙΟ ΣΤΟ ΜΟΝΟΠΑΤΙ

Στο 8^{ο} κεφάλαιο αυτής της πτυχιακής, θα περιγραφούν βήμα βήμα οι ενέργειες του ρομπότ όταν αυτό εντοπίσει ένα εμπόδιο στο μονοπάτι, ώστε να το αποφύγει και να συνεχίσει την περιπολεία.

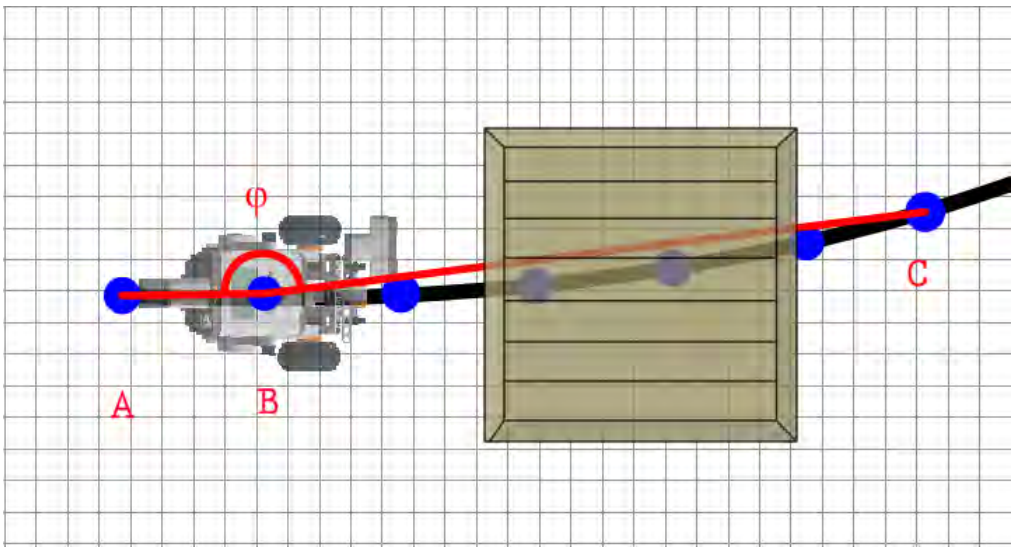
8.1 ***Εμπόδιο εντοπίστηκε***

Όταν ο αισθητήρας απόστασης βρει εμπόδιο μέσα στην εμβέλεια του, θέτει ως παρούσα θέση τη θέση που βρίσκετα, αποθηκεύει τις συνταταγμένες του σημείου αυτού και καλέι τη συνάρτηση ***overtake()***.

```
{
    pose.setLocation(xy[i][0], xy[i][1]);
    overTake();

}
```

8.2 ***Η συνάρτηση overtake()***

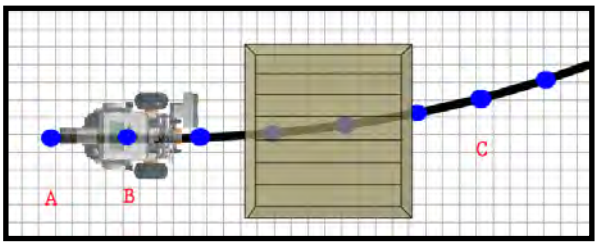
Μόλις κληθεί η συνάρτηση, βρίσκει την γωνία φ που σχηματίζεται από τις ευθείες AB και BC που φαίνονται στο σχήμα παρακάτω:

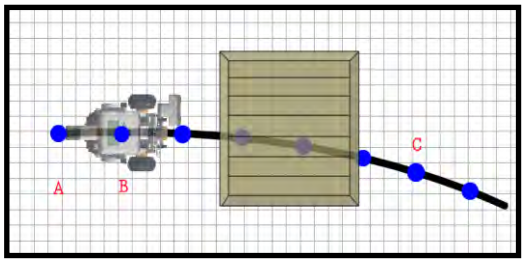


Με αυτόν τον τρόπο το ρομπότ καταλαβαίνει προς τα πού είναι πιθανό να συνεχίζει το μονοπάτι και στρίβει προς αυτήν την πλευρά για να αποφύγει το εμπόδιο. Αν η γωνία είναι από μηδέν μοίρες και πάνω στρίβει προς τα δεξιά, αλλιώς στρίβει αριστερά.

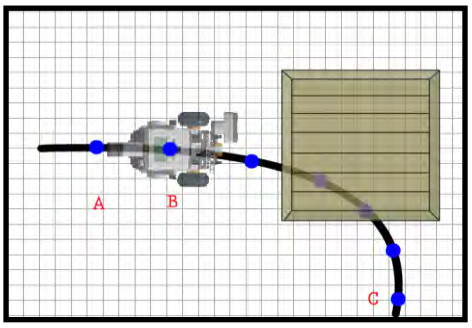
Στη συνέχεια αναλόγως πού έχει στρίψει εξετάζει τις παρακάτω περιπτώσεις:

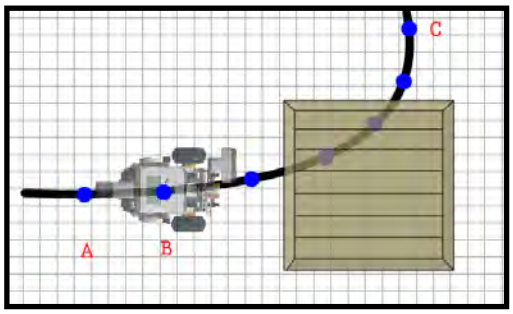
- Το μονοπάτι διαπερνά το εμπόδιο και εξέρχεται από την απέναντι μεριά του



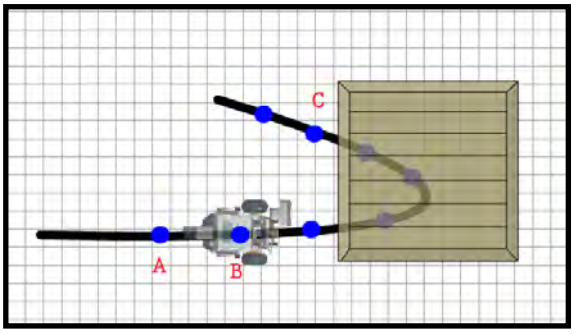


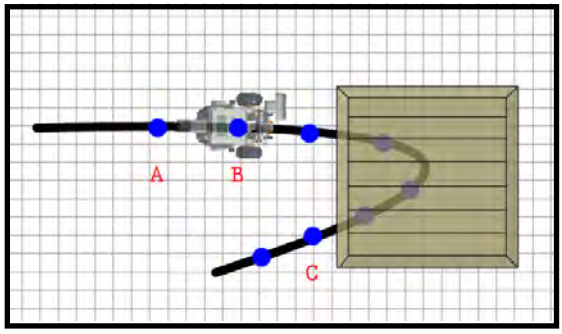
- Το μονοπάτι διαπερνά το εμπόδιο και εξέρχεται από τη μία πλαϊνή πλευρά του





- Το μονοπάτι εισέρχεται στο εμπόδιο και εξέρχεται από την ίδια πλευρά που βρίσκεται το ρομπότ.





8.2.1 <u>Η πρώτη περίπτωση</u>

Εδώ το μονοπάτι εξέρχεται από την απέναντι πλευρά του εμποδίου. Το ρομπότ μας καλείται να στρίψει στην σωστή μεριά, να κινηθεί κάθετο στο εμπόδιο έως ότου δεν το βλέπει και να στρίψει ώστε να βρίσκεται παράλληλα με το μονοπάτι ξανά. Εκεί υπολογίζει το σημείο που βρίσκεται με τη βοήθεια της κλάσης **Pose()** και κάνοντας χρήση των μεθόδων της, **getPose().getX()** και **getPose().getY()**

αποθηκεύουμε τις συντεταγμένες του σημείου. Στη συνέχεια, προχωράει παράλληλα στο μονοπάτι έως ότου πάλι να μη βλέπει το εμπόδιο. Για ακόμη μία φορά υπολογίζει τις συντεταγμένες του σημείου που βρίσκεται και έπειτα διαπερνά τον πίνακα με τις συνταγμένες, που έχουμε στείλει από τον υπολογιστή, από το σημείο που συνάντησε το εμπόδιο μέχρι τέλος και βρίσκει από ποιο σημείο απέχει τη λιγότερη απόσταση και κινείται πρός αυτό. Τέλος στρίβει ξανά προς την κατεύθυνση που συνεχίζει το μονοπάτι και συνεχίζει την περιπολεία.

8.2.2 <u>Η δεύτερη περίπτωση</u>

Σε αυτήν την περίπτωση το μονοπάτι εξέρχεται από μία από τις πλευρές του εμποδίου. Αυτό σημαίνει ότι το ρομπότ θα εκτελέσει τις παραπάνω λειτουργίες αλλά θα σταματήσει ένα βήμα πιο νωρίς. Πιο αναλυτικά, το ρομπότ στρίβει προς την αντίστοιχη πλευρά που θα κινηθεί το μονοπάτι και κινείται μέχρι ο αισθητήρας απόστασης να μη βλέπει πια το εμπόδιο. Βρίσκει το σημείο που έχει σταματήσει το ρομπότ και αφού στρίψει ώστε να είναι παράλληλα με το μονοπάτι, κινείται προς το σημείο με τη μικρότερη απόσταση.

8.2.3 <u>Η τρίτη περίπτωση</u>

Εδώ το μονοπάτι κάνει στροφή που φτάνει τις 180 μοίρες. Εξέρχεται δηλαδή από την ίδια πλευρά που βρίσκεται και το ρομπότ μας. Για να βρει το σημείο στο οποίο πρέπει να συνεχίσει τη διαδρομή, αυτό που έχει να κάνει είναι να στρίψει αρχικά προς τη σωστή πλευρά να υπολογίσει ποιο είναι το σημείο που έχει τη μικρότερη απόσταση και βρίσκεται έξω από το εμπόδιο και κινείται προς αυτό. Τέλος στρίβει προς τη σωστή φορά και συνεχίζει την περιπολεία.

9. ΣΥΜΠΕΡΑΣΜΑΤΑ

Σε αυτό το κεφάλαιο θα παρατέθουν τα προβλήματα που αντιμετωπίσαμε κατά τη διάρκεια αυτής της διατριβής αλλά ακόμα και πριν την αρχή της. Θα προταθούν βελτιώσεις όσον αφορά την κατασκευή του ρομπότ αλλά και στον κώδικα. Επίσης θα αναφερθούμε και σε πιθανές μελλοντικές χρήσεις.

9.1 Προβλήματα-Περιορισμοί

Το πρόβλημα εξ' αιτίας του οποίου ο κώδικας της πτυχιακής κατασκευάστηκε έτσι αφορά το Bluetooth. Παρά τις εκτεταμένες μας προσπάθειες να επικοινωνήσουν ο υπολογιστής με το Lego NXT δεν τα καταφέραμε. Το αξιοπερίεργο είναι πως ο υπολογιστής μπορούσε μέσω του Bluetooth να εντοπίσει και να εξερευνήσει τα αρχεία του τούβλου NXT, παρ' όλα αύτα όταν θέλαμε να ανεβάσουμε το πρόγραμμα, μέσω του Eclipse αυτή τη φορά, δεν ήταν δυνατή η επικοινωνία τους.

Αυτό οδήγησε στο να κινηθούμε με γνώμωνα το καλώδιο USB, που με τη σειρά του έφερε στο φως αναμενόμενα προβλήματα. Ένα από αυτά ήταν ο αποθηκευτικός χώρος στο ρομπότ, καθώς έπρεπε να του στείλουμε όλες τις πληροφορίες από τον υπολογιστή. Οι αρχικά μεγάλοι πίνακες με δεδομένα μας προκαλούσαν σφάλμα από μεριάς του NXT επειδή καταλάμβαναν όλο τον αποθηκευτικό του χώρο, χωρίς καν να περαστούν όλα τα στοιχεία. Έτσι χρησιμοποιήσαμε δυναμικούς πίνακες. Αυτό θα μπορούσε εύκολα να επιλυθεί αν δούλευε το Bluetooth οπότε θα μπορούσαμε να στέλνουμε μικρά κομμάτια δεδομένων τη φορά.

Ένα άλλο πρόβλημα με την έλλειψη Bluetooth, ήταν κατά τη διάρκεια εντοπισμού εμποδίου. Με τη χρήση Bluetooth, θα μπορούσαμε να στέλνουμε σήμα απο το ρομπότ στον υπολογιστή πως εντοπίστηκε εμπόδιο στο μονοπάτι και ο υπολογισμός για την αποφυγή του να γίνει στον υπολογιστή και μόλις βρει τις κατάλληλες κινήσεις για το ρομποτ ώστε να προσπεράσει το εμπόδιο, να του στείλει τα δεδομένα.

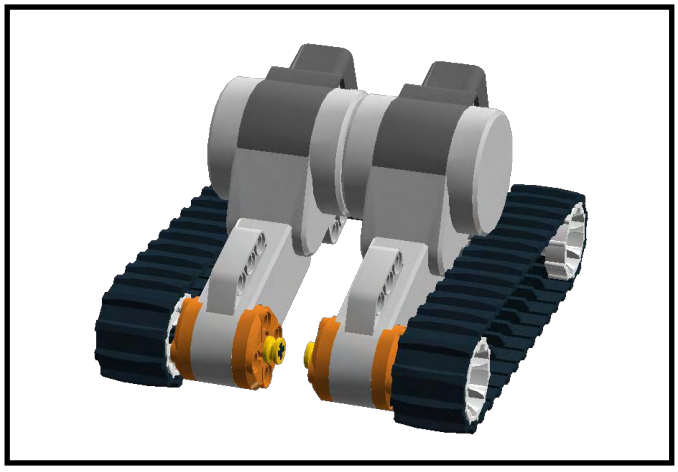
Επίσης, τα κομμάτια που είχα στη διάθεσή μου για να κατασκευάσω το ρομπότ ήταν περιορισμένα. Αυτό είχε ως αποτέλεσμα να βασιστώ σε μεγάλο βαθμό στη φαντασία μου, να βρω τρόπο ώστε η κατασκευή μου να πληρεί τις προδιαγραφές για το σκοπό που θέλαμε να φέρει εις πέρας. Στη διάθεσή μας είχαμε τρεις σερβοκινητήρες. Δύο από αυτούς ήταν απαραίτητοι για να επιτευχθούν κινήσεις στο επίπεδο (εμπρός, πίσω και να μπορεί να στρίβει δεξιά και αριστερά). Ο άλλος κινητήρας χρειαζόταν ώστε να στρέφει τον αισθητήρα και να μπορούμε να εντοπίζουμε το εμπόδιο στο μονοπάτι και να μετρήσουμε τις πλευρές του. Επομένως αναλόγως προς τα πού θα έστριβε έπρεπε και να στραφεί ο αισθητήρας αντίστοιχα.

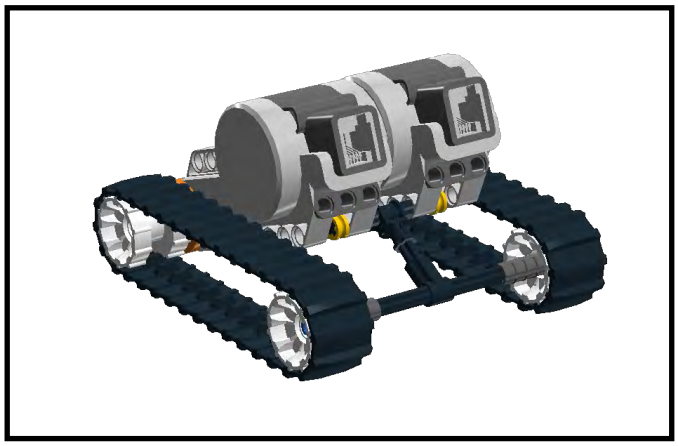
Ακόμα ο αισθητήρας ήχου έπρεπε να είναι σε μία μικρή απόσταση από τους κινητήρες. Αυτό έπρεπε να ληφθεί υπ' όψιν γιατί, οι κινητήρες όταν κινούνται βγάζουν το χαρακτηριστικό ήχο των μοτέρ και ήταν αυτός ήταν αρκετός για να θεωρήσει το ρομπότ μας πως είναι ο ήχος λήξης.

9.2 Βελτιώσεις

Όπως αναφέρφηκε προηγουμένως, εφόσον λειτουργούσε το Bluetooth θα μπορούσε ο κώδικας μου στέλνουμε στο ρομπότ να είναι πιο εκλεπτυσμένος και ο όγκος δεδομένων πολύ λιγότερος.

Ακόμη, ενώ η κατασκευή, δομικά, συμπεριφέρεται άψογα, θα μπορούσαμε με πληθώρα κομματιών Lego να ανακατασκευάσουμε το ρομπότ ώστε αντί οι δυο κινητήρες να κινούν δύο τροχούς, να μπουν αντίθετα ο ένας από τον άλλον και ο καθένας να κινεί δύο μικρότερους τροχους που αυτοί με τη σειρά τους θα είναι συνδεδεμένοι ανά δύο με τις λαστιχιένιες ερπύστριες που είναι κομμάτι των Lego Mindstorms, όπως φαίνεται ένα παράδειγμα στις εικόνες παρακάτω. Αυτό θα μας επέτρεπε μεγαλύτερη ευελιξία στις κινήσεις.





Θα μπορούσε επίσης να υποστηρίζει κι άλλους αισθητήρες ή ακόμα και μία υποδοχή για Smartphone με σκοπό να εκμεταλευτούμε την κάμερά του και άλλες εφαρμογές (π.χ. GPS).

9.3 Μελλοντικές χρήσεις

Το παρών ρομπότ εκτελεί περιπολίες πάνω σε ένα μονοπάτι που έχει εισάγει ο χρήστης από τον υπολογιστή και καθ' όλη τη διαρκεια ελέγχει για εμπόδια. Με τις κατάλληλες διαμορφώσεις και στον κώδικα αλλά και στο ρομπότ θα μπορούσε να χρησιμοποιηθεί στον έλεγχο δρόμων από πόλη σε πόλη για πιθανά εμπόδια ή πρόσφατες κατολισθήσεις, θα καταγράφει την τοποθεσία τους και θα στέλνει σήμα μέσω ασύρματου δικτύου, για να σπεύσουν κατάλληλες υπηρεσίες για την απομάκρυνση του εμποδίου.

Θα μπορούσε ακόμα να χρησιμοποιηθεί στα έξυπνα σπίτια σαν επείγουσα βοήθεια για τους κατοίκους σε περίπτωση ανάγκης ή ακόμα και όταν κοιμούνται οι ένοικοι το ρομπότ θα περιπολεί μέσα στο σπίτι, θα ελέγχει για εισβολείς και θα στέλνει τα δεδομένα που θα λαμβάνει η κάμερά του, στο server του σπιτιού.

Επιπλέον, χρησιμοποιώντας το ρομπότ σαν 'μακέτα' θα μπορούσε να βοηθήσει στη δημιουργία αυτόματων φορείων σε νοσοκομεία για τη μεταφορά ασθενών από το ένα τμήμα του νοσοκομείου στο άλλο, χωρίς να μειώνεται ο αριθμός του ενεργού ιατρικού προσωπικού.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1.**Εργαστήριο Ρομποτικής - Ρομποτική**: http://users.sch.gr/jenyk/index.php/robotics

2.**Εργαστήριο Ρομποτικής - Ιστορική Αναδρομή**:
http://users.sch.gr/jenyk/index.php/robotics/robotics-historicalreview/37-talos

3. **3ο Γενικό Λύκειο Πτολεμαΐδας - Ιστορική Ανδρομή**:
http://3ogelptolrobot.weebly.com/piomicroniotaalpha-epsilon943nualphaiota-eta-iotasigmatauomicronrho943alpha-tauomicronupsilon-rhoomicronmupi972tau--tauetasigmaf-rhoomicronmupiomicrontauiotakappa942sigmaf.html

4. **Εργαστήριο Ρομποτικής - Οι 3 νόμοι των Ρομπότ**:
http://users.sch.gr/jenyk/index.php/robotics/robotics-historicalreview/28-the3rulesofrobotics

5. **Robot Hall Of Fame**: http://www.robothalloffame.org/inductees/03inductees/unimate.html

6. **Artificial Intelligence Center**: http://www.ai.sri.com/shakey/

7. **LEGO**: http://aboutus.lego.com/en-us/lego-group/the_lego_history

8. **LEGO - Mindstorms**: http://www.lego.com/en-us/mindstorms/gettingstarted/historypage/

9. **Wikipedia - Lego Mindstorms**: http://el.wikipedia.org/wiki/Lego_Mindstorms

10. *Κώστας Δημητρίου, Γιώργος Κορρές* - ***Εισαγωγή στον προγραμματισμό με τα Lego Mindstorms NXT*** (σ. 232). Διερευνητική Μάθηση Α. Ε.

11. **Lego Engineering - NXT G**: http://www.legoengineering.com/program/nxt-g/

12. **Next Byte Codes, Not eXactly C, and SuperPro C**: http://bricxcc.sourceforge.net/nbc/

13. **Lejos**: http://www.lejos.org/

14. **MathWorks**: http://www.mathworks.com/hardware-support/lego-mindstorms-matlab.html

15. **Microsoft**: https://www.microsoft.com/en-us/download/details.aspx?id=29081

16. **Microsoft Dreamspark**: https://www.dreamspark.com/Product/Product.aspx?productid=30

17. **Robot C**: http://www.robotc.net/

18. **nxt-python**: https://code.google.com/p/nxt-python/

19. **Lego - Lego Digital Designer**: http://ldd.lego.com/en-us/

ΠΑΡΑΡΤΗΜΑ Α

Παρατείθεται ο κώδικας του προγράμματος παρακάτω, αρχίζοντας από **το κομμάτι του υπολογιστή**:

Main.java

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.applet.*;
import java.util.Vector;



public class Main extends Applet {
	public void init() {
	Path.count=1; // Αρχικοποιούμε τη μεταβλητή count στην κλάση Path

	// Αρχικοποίηση του διδιάστατου πίνακα που θα αποθηκευθούν
	// τα σημεία του μονοπατιού
	for(int i=0; i<400; i++)
	{
		for (int j=0; j<400; j++)
		{
			Path.coords[i][j]=0;
		}
	}
	setLayout(new BorderLayout());
	DrawPanel dp = new DrawPanel();  // Δημιουργούμε ένα αντικείμενο dp
τύπου DrawPanel
	add("Center", dp);                              // στην οποία ορίζουμε το
κέντρο της
	add("South",new DrawControls(dp));  // και το σημείο που θα βρίσκονται
τα πλήκτρα

	}

	// Συνθήκη για το κλείσιμο του παραθύρου
    public boolean handleEvent(Event e) {
      switch (e.id) {
        case Event.WINDOW_DESTROY:
          System.exit(0);
          return true;
        default:
          return false;
      }
    }

    public static void main(String args[]) {
      Frame f = new Frame("Canvas"); // Δημιουργούμε ένα αντικείμενο f τύπου
Frame με το όνομα Canvas
```

```
        Main drawTest = new Main(); // Δημιουργούμε ένα αντικείμενο drawTest
τύπου Main
        drawTest.init(); // Καλούμε τη μέθοδο init() για να φορτώσουμε το applet
        drawTest.start(); // Καλούμε τη μέθοδο start() για να ξεκινήσουμε το
applet

        f.add("Center", drawTest); // Ορίζουμε το κέντρο στο παράθυρό μας
        f.setSize(400, 400);         // του δίνουμε μέγεθος 400 * 400 pixels
        f.setResizable(false);       // αφαιρούμε τη δυνατότητα αλλαγής
μεγέθους του παραθύρου
        f.setVisible(true);          // και το ορίζουμε ως "εμφανές" δηλαδή να
φανεί στην οθόνη του
    }                                       // υπολογιστή
}

class DrawPanel extends Panel {

    //public static final int LINES = 0;
    public static final int POINTS = 1; // Δήλωση μεταβλητής POINTS και
καταχωρούμε την τιμή 1
    int      mode = POINTS; // Δήλωση μεταβλητής mode που καταχωρούμε την
POINTS γιατί θέουμε να χαράζουμε το μονοπάτι pixel προς pixel
    Vector lines = new Vector(); // Δήλωση πολυδιάστατου πίνακα lines που θα
αποθηκεύονται τα σημεία
    Vector colors = new Vector(); // Δήλωση πολυ διάστατου πίνακα colors που
θα έχουμε καταχωρημένα τα χρώματα με τα οποία θα χαράζουμε τα pixels. Στην
προκειμένη το μάυρο.
    // Δήλωση μερικών βοηθητικών μεταβλητών
    int x1,y1;
    int x2,y2;
    int xl, yl;

    public DrawPanel() {
       setBackground(Color.white); // Ορίζουμε το φόντο του DrawPanel άσπρο
    }

    // Μέθοδος η οποία ορίζει τον τρόπο που θα χαράξουμε τα pixels, βάσει της
επιλογής μας από το dropdown μενού
    public void setDrawMode(int mode) {
       switch (mode) {
        // case LINES:
         case POINTS:
           this.mode = mode;
           break;
         default:
           throw new IllegalArgumentException();
       }
    }

    // Στη handleEvent ορίζουμε τι θα γίνει στις παρακάτω περιπτώσεις
    public boolean handleEvent(Event e) {
       switch (e.id) {
         case Event.MOUSE_DOWN: // Μόλις πατηθεί το κουμπί του ποντικιού
           switch (mode) {
             case POINTS:
```

```
            default:
             colors.addElement(getForeground()); // Χρησιμοποιώντας το χρώμα
μαύρο
             lines.addElement(new Rectangle(e.x, e.y, -1, -1)); // προσθέτει
στον πίνακα lines τις συντεταγμένες x και y που βρίσκεται ο κέρσορας
             x1 = e.x; // Επίσης τα καταχωρούμε στις x1 και y1
             y1 = e.y;
             repaint(); // Και καλώντας την repaint() ενημερώνουμε τη μέθοδο
paint() που θα δούμε παρακάτω, για τις αλλαγές
             break;
          }
          return true;
        case Event.MOUSE_UP: // Μόλις αφήσουμε το πλήκτρο του ποντικιού
          switch (mode) {
            case POINTS:
            default:
             break;
          }
          repaint();
          return true;
        case Event.MOUSE_DRAG: // Αν σύρουμε το ποντίκι
          switch (mode) {
            case POINTS:
            default:
             colors.addElement(getForeground());
             lines.addElement(new Rectangle(x1, y1, e.x, e.y));
             x1 = e.x;
             y1 = e.y;

             if (((x1>=0) || (x1<400)) && ((y1>=0) || (y1<400))) // Έλεγχος
για να μη βγούμε εκτός ορίων του πίνακα
             Path.coords[x1][y1]=Path.count++; // Αυξάνουμε την count

             break;
          }
          repaint();
          return true;
        case Event.WINDOW_DESTROY:
              System.exit(0);
          return true;
        default:
          return false;
      }
    }

    public void paint(Graphics g) {
      int np = lines.size(); // Παίρνουμε το μέγεθος του πίνακα lines

      // Ζωγραφίζουμε τα pixels μας
      g.setColor(getForeground());
      g.setPaintMode();

      // Ένα loop για να περάσουμε όλα τα στοιχεία του πίνακα lines
      for (int i=0; i < np; i++) {
          Rectangle p = (Rectangle)lines.elementAt(i);
```

```
            g.setColor((Color)colors.elementAt(i));
            if (p.width != -1) {
               g.drawLine(p.x, p.y, p.width, p.height);
            } else {
               g.drawLine(p.x, p.y, p.x, p.y);
            }
         }

         }
}

// Εδώ ορίζουμε τα συστατικά μέρη του γραφικού περιβάλλοντος
class DrawControls extends Panel {
    DrawPanel target;

    public DrawControls(DrawPanel target) {
       this.target = target;
       setLayout(new FlowLayout());
       setBackground(Color.CYAN); // Το χρώμα του παραθύρου
       Choice shapes = new Choice(); // Δημιουργία αντικειμένου shapes τύπου
Choice (dropdown μενού)
       shapes.addItem("Points"); // Προσθέτουμε στο αντικείμενο το Points
       shapes.setBackground(Color.lightGray); // Ορίζουμε το χρώμα του φόντου
του

       final Button SendAgain = new Button("Send Again"); // Δημηουργία του
πλήκτρου SendAgain
       SendAgain.setPreferredSize(new Dimension(80, 30)); // Ορίζουμε το
μέγεθός του
       SendAgain.setEnabled(false); // Αρχικά θέλουμε να είναι απενεργοποιημένο
       final Button Send = new Button("Send Data");  // Δημηουργία του πλήκτρου
Send
       Send.setPreferredSize(new Dimension(80, 30)); // Ορίζουμε το μέγεθός του
       Button Exit = new Button("Exit"); // Δημηουργία του πλήκτρου Exit
       Exit.setPreferredSize(new Dimension(80,30)); // Ορίζουμε το μέγεθός του
       add(SendAgain); // Προσθέτουμε στο DrawControls το SendAgain
       add(Send); // Προσθέτουμε στο DrawControls το Send
       add(Exit); // Προσθέτουμε στο DrawControls το Exit
       add(shapes); // Προσθέτουμε στο DrawControls το shapes




       Send.addActionListener(new ActionListener() // Ορισμός λειτουργιών του
Send
       {
              public void actionPerformed(ActionEvent e)
              {
                     Path.PrintMap("c:\\map.txt"); // Κάλεσε την PrintMap από
την Path με όρισμα "c:\\map.txt"
              Path.BuildTable(); // Κάλεσε την BuildTable από την Path
              SendAgain.setEnabled(true); // Ενεργοποίησε το SendAgain
              Send.setEnabled(false); // Απενεργοποίσε τον εαυτό σου
              }
```

```
      });

      SendAgain.addActionListener(new ActionListener() // Ορισμός λειτουργιών
του SendAgain
      {
            public void actionPerformed(ActionEvent e)
            {
                  SimplifiedPath.CalculatePath(); // Κάλεσε την
CalculatePath από την SimplifiedPath
            }

      });

      Exit.addActionListener(new ActionListener() // Ορισμός λειτουργιών του
Exit
      {
            public void actionPerformed(ActionEvent e)
            {
                  System.exit(0); // Κλείσε το παράθυρο
            }

      });

   }

   public void paint(Graphics g) {
      Rectangle r = bounds();
      g.setColor(Color.lightGray);
      g.draw3DRect(0, 0, r.width, r.height, false);
   }

   public boolean action(Event e, Object arg)
   {
         {
            String choice = (String)arg;

                  if (choice.equals("Points"))
                  {
                        target.setDrawMode(DrawPanel.POINTS);
                  }
            }
          return true;
   }
}
```

Τέλος Main.java

Path.java

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;


public class Path {
	static int size=400; // Αρχικοποίηση της μεταβλητής size
	static int x=size;   // Αρχικοποίηση της μεταβλητής x
	static int y=size;   // Αρχικοποίηση της μεταβλητής y
	static int count;    // Δημιουργία της μεταβλητής count
	static ArrayList<int[]> pathIndexD = new ArrayList<int[]>(); //
Δημιουργία του δυναμικού διδιάστατου
	// πίνακα pathIndexD με δυναμικό μέγεθος γραμμών και πλήθος στηλών 2


	static int[][] coords = new int[x][y]; // Δημιουργία διδιάστατου πίνακα
με μέγεθος x * y

	/*public static void show()
	{
		for(int i=0; i<size; i++)
		{
			for (int j=0; j<size; j++)
			{
				System.out.println(Path.coords[i][j]);
			}
		}
	}*/

	// Με την παρακάτω συνάρτηση εκτυπώνουμε σε ένα αρχείο txt
	// το μονοπάτι που χαράξαμε
	static void PrintMap(String filename)
	{
		try
		{
			FileWriter f = new FileWriter(filename); // Δημιουργούμε
ένα αντικείμενο f τύπου FileWriter
			BufferedWriter b = new BufferedWriter(f); // Δημιουργούμε
ένα αντικείμενο b τύπου BufferedWriter

			for (int i = 0; i < size; i++)
			{
				for(int j=0; j< size; j++)
				{
				if (coords[j][i]>=1) // Εξετάζουμε αν στον πίνακα
coords το στοιχείο που προσπελάυνει
                                                    // είναι
μεγαλύτερο ή ίσο του 1
                                b.write(String.valueOf(coords[j][i])); //και
αν είναι τότε το εκτυπώνει
				else
```

```
                                    b.write((char)32); // αλλιώς βάζει στο txt
τον κενό χαρακτήρα
                             }
                             b.newLine(); // όταν τελειώσει μία γραμμή, αλλάζει
σειρά
                      }
                      b.close(); // τέλος κλείνει το stream και αδειάζει τη
μνήμη
               }
               catch (IOException e)
               {
                      System.out.println(e);
               }
        }
        // Στην παρακάτω συνάρτηση γεμίζουμε τον πίνακα pathIndexD
        public static void BuildTable()
        {
               int k=1; // Αρχικοποιούμε τη μεταβλητή k
               while (k<count)
               {
                      int flag=0; // Αρχικοποιούμε μια βοηθητική μεταβλητή flag
               for (int i = 0; i < size; i++) // Το i αντιπροσωπεύει τα x
               {
                      for(int j = 0; j < size; j++) // Το j αντιπροσωπεύει τα y
                      {
                             if (flag==1) // Αν η flag είναι ίση με 1
                             break;       // βγες από το loop
                             if (coords[i][j] == k) // Αν το στοιχείο από τον
πίνακα coords είναι ίσο
                             // με το k τότε:
                             {

                                    pathIndexD.add(new int[]{i,size-j}); //
Πρόσθεσε τις συντεταγμένες στον πίνακα pathIndexD
                                    System.out.print(pathIndexD.get(k-1)[0]);
                                    System.out.println(" "+pathIndexD.get(k-
1)[1]);
                                    k++; // Αύξησε το k κατα ένα
                                    flag=1; // και θέσε το flag=1 ώστε να ξέρουμε
ότι ελέγξαμε αυτό το σημείο
                                    break;
                             }
                      }
               }
               }

               SimplifiedPath SP = new SimplifiedPath(); // Δημιουργούμε ένα
αντικείμενο SP τύπου SimplifiedPath
               SP.CalculatePath(); // Καλούμε την μέθοδο CalculatePath από το SP
        }

}
```

Τέλος Path.java

SimplifiedPath.java

```
import java.awt.geom.Line2D;


public class SimplifiedPath
{
	final static double wheelDiam = 5.6; // Διάμετρος ρόδας
	final static double wheelDist = 12.5; // Απόσταση ροδών

	// Συνάρτηση υπολογισμού της γωνίας ανάμεσα σε δύο ευθείες που δίνουν ως
ορίσματα
	public static int angleBetween2Lines(Line2D line1, Line2D line2)
    {
		// Υπολογισμός της γωνίας που σχηματίζεται με τις πολικές
συντεταγμένες της μίας ευθείας, με τη χρήση της Math.atan2
		int angle1 = (int) (180/ Math.PI *
(Math.atan2(Math.abs(line1.getY1() - line1.getY2()),
		   Math.abs(line1.getX1() - line1.getX2()))));
		// Υπολογισμός της γωνίας που σχηματίζεται με τις πολικές
συντεταγμένες της δεύτερης ευθείας, με τη χρήση της Math.atan2
		int angle2 = (int) (180/ Math.PI *
(Math.atan2(Math.abs(line2.getY1() - line2.getY2()),
		   Math.abs(line2.getX1() - line2.getX2()))));
		// Αφαιρώντας τα παραπάνώ αποτελέσματα βρίσκουμε τη γωνία ανάμεσα
στις δύο ευθείες
		return angle1-angle2;
    }
	// Συνάρτηση για τον υπολογισμό των ευθειών και των γωνιών που
σχηματίζονται από τα σημεία που χαράξαμε
	public static void CalculatePath()
	{
		double pie=Math.PI; // π = 3,14
		double WheelPerim = wheelDiam*pie; // Περίμετρος ρόδας
		double r = wheelDiam/2; // Ακτίνα ρόδας
		int currentLine; // Δημιουργία μεταβλητής για της τρέχουσα γραμμή
		int currentAngle; // Δημιουργία μετααβλητής για την τρέχουσα
γωνία

		int cnt=0;
		int[] segmentsInWheelDegrees = new int[Path.pathIndexD.size()-1];
// Δημιουργία μονοδιάστατου πίνακα για τα μήκη των ευθειών
		int[] anglesInWheelDegrees = new int[Path.pathIndexD.size()-2];
// Δημιουργία μονοδιάστατου πίνακα για τις γωνίες
		int xcoord=-1;
		int ycoord=-1;

		// Γεμίζουμε τον πίνακα segmentsInWheelDegrees
		while (cnt<segmentsInWheelDegrees.length)
		{
			xcoord=(Path.pathIndexD.get(cnt)[0])-
(Path.pathIndexD.get(cnt+1)[0]);
			ycoord=(Path.pathIndexD.get(cnt)[1])-
(Path.pathIndexD.get(cnt+1)[1]);
			// ypologismos mhkous meta3y dyo shmeiwn (pixels) //
```

```
                    currentLine= (int) Math.sqrt
(Math.pow(xcoord,2)+Math.pow(ycoord, 2));

      segmentsInWheelDegrees[cnt]=(int)((currentLine*360)/(pie*wheelDiam));
                    System.out.print(cnt + " ");
                    System.out.println(segmentsInWheelDegrees[cnt]);

                    cnt++;
             }
             cnt=0;
             // Γεμίζουμε τον πίνακα anglesInWheelDegrees
             while (cnt<anglesInWheelDegrees.length)
             {
                    // Υπολογισμός δύο διαδοχικών ευθειών
                    Line2D line1 = new Line2D.Float();
                    Line2D line2 = new Line2D.Float();
                    line1.setLine(Path.pathIndexD.get(cnt)[0],
Path.pathIndexD.get(cnt)[1], Path.pathIndexD.get(cnt+1)[0],
Path.pathIndexD.get(cnt+1)[1]);
                    line2.setLine(Path.pathIndexD.get(cnt+1)[0],
Path.pathIndexD.get(cnt+1)[1], Path.pathIndexD.get(cnt+2)[0],
Path.pathIndexD.get(cnt+2)[1]);

                    // Κλήση συνάρτησης angleBetween2Lines που θα μας
επιστρέψει τη γωνία μεταξύ των ευθειών
                    currentAngle=angleBetween2Lines(line1, line2);

                    // Μετατρέπουμε τη γωνία σε μοίρες περιστροφής ρόδας
                    anglesInWheelDegrees[cnt]=(int)
((currentAngle*2*pie*wheelDist)/WheelPerim);

                    System.out.print(cnt + " ");
                    System.out.println(anglesInWheelDegrees[cnt] +" ");
                    cnt++;

             }
                    System.out.println("size of segments" +
segmentsInWheelDegrees.length);
                    System.out.print("size of angles" +
anglesInWheelDegrees.length);
             // Κλήση της μεθόδου comms της κλάσης SendData, με ορίσματα τον
διδιαστατο δυναμικό πίνακα pathIndexD,
             // τον πίνακα με τα μήκη των ευθειών segmentsInWheelDegrees και
τον πίνακα με τις περιστροφές ρόδας
             // για τις γωνίες anglesInWheelDegrees
             SendData.comms(Path.pathIndexD, segmentsInWheelDegrees,
anglesInWheelDegrees);
      }

}
```

Τέλος SimplifiedPath.java

SendData.java

```
import lejos.pc.comm.*;
import java.io.*;
import java.util.ArrayList;


public class SendData {
	public static void comms(ArrayList<int[]> coords, int[] segments, int[]
angles) {
		NXTConnector conn = new NXTConnector();

		conn.addLogListener(new NXTCommLogListener(){

			public void logEvent(String message) {
				System.out.println("USBSend Log.listener:
"+message);


			}

			public void logEvent(Throwable throwable) {
				System.out.println("USBSend Log.listener - stack
trace: ");
				 throwable.printStackTrace();

			}

		}
		);

		if (!conn.connectTo("NXT#1")){
			System.err.println("No NXT found using USB");
			System.exit(1);
		}

		//DataInputStream inDat = conn.getDataIn();
		DataOutputStream outDat = conn.getDataOut(); // Δημιουργία
αντικειμένου outDat που θα χρησιμοποιηθεί για την απόστολη δεδομένων στο NXT


		try {
			outDat.writeInt(Path.pathIndexD.size()); // Αποστολή του
μεγέθους του pathIndexD
			outDat.flush(); // Άδειασμα της μνήμης
			try { Thread.sleep(1000); } catch (InterruptedException e)
{}
		} catch (IOException ioe) {
			System.err.println("IO Exception writing bytes");
		}

		for(int i=0;i<coords.size();i++)
		{
			try {
			   outDat.writeInt(coords.get(i)[0]); // Αποστολή των x
συντεταγμένων του δυναμικού πίνακα coords(pathIndexD)
```

```
                        outDat.flush();
                        try { Thread.sleep(100); } catch (InterruptedException
e) {}

                    } catch (IOException ioe) {
                        System.err.println("IO Exception writing bytes");
                    }
                }
                for(int i=0;i<coords.size();i++)
                {
                    try {
                        outDat.writeInt(coords.get(i)[1]); // Αποστολή των y
συντεταγμένων του δυναμικού πίνακα coords(pathIndexD)
                        outDat.flush();
                        try { Thread.sleep(100); } catch (InterruptedException
e) {}

                    } catch (IOException ioe) {
                        System.err.println("IO Exception writing bytes");
                    }
                }
                for(int i=0;i<segments.length;i++)
                {
                    try {
                        outDat.writeInt(segments[i]); // Αποστολή του πίνακα
segments(segmentsInWheelDegrees) που περιέχει τις ευθείες
                        outDat.flush();
                        try { Thread.sleep(100); } catch (InterruptedException
e) {}

                    } catch (IOException ioe) {
                        System.err.println("IO Exception writing bytes");
                    }
                }
                for(int i=0;i<angles.length;i++)
                {
                    try {
                        outDat.writeInt(angles[i]); // Αποστολή του πίνακα
angles(anglesInWheelDegrees) που περιέχει τις γωνίες σε περιστροφές ρόδας
                        outDat.flush();
                        try { Thread.sleep(100); } catch (InterruptedException
e) {}

                    } catch (IOException ioe) {
                        System.err.println("IO Exception writing bytes");
                    }
                }

                try {
                    //inDat.close();
                    outDat.close(); // Κλείνουμε το stream
                    System.out.println("Closed data streams");
                } catch (IOException ioe) {
                    System.err.println("IO Exception Closing connection");
                }
```

```
            try {
                  conn.close(); // Κλενουμε τη σύνδεση
                  System.out.println("Closed connection");
            } catch (IOException ioe) {
                  System.err.println("IO Exception Closing connection");
            }
            }
}
```

Τέλος SendData.java

Εδώ ολοκληρώνεται πρόγραμμα που δημιουργεί το γραφικό περιβάλλον στον υπολογιστή. Στη συνέχεια δίνεται ο κώδικας που αφόρα το κομμάτι του ρομπότ Lego NXT.

Connection.java

```
import java.io.*;
import lejos.nxt.*;
import lejos.nxt.comm.*;


public class Connection {

	public static void main(String[] args) {

		LCD.drawString("Press Right",0, 0); // Εμφάνιση στην οθόνη του
NXT το μήνυμα
		LCD.drawString("for BT",0,1);		// "Press Right for BT"
	    NXTConnection connection = null;

	    // Αν πατηθεί το δεξιά κουμπί, τότε το NXT περιμένει δεδομένα μέσω
Bluetooth
	    if(Button.waitForPress() == Button.ID_RIGHT){
	      LCD.clear();
	      LCD.drawString("waiting for BT", 0,0 );
	      connection = Bluetooth.waitForConnection();
	    }
	    // Αλλιώς περιμένει δεδομένα μέσω USB
	    else {
	      LCD.clear();
	      LCD.drawString("waiting for USB", 0,0 );
	      connection = USB.waitForConnection();
	    }
	    DataInputStream dataIn = connection.openDataInputStream(); //
Δημιουργία αντικειμένου dataIn που θα χρησιμοποιηθεί για την απόδοχή δεδομένων
από τον υπολογιστή
	    int length=0; // Αρχικοποίηση της μεταβλητής length
	    int i=0; // Αρχικοποίηση της μεταβλητής i
	    try {
			length=dataIn.readInt(); // Καταχωρούμε στη length το
πρώτο πράγμα που λαμβάνουμε, δηλαδή το μέγεθος του pathIndexD
			LCD.drawString("Length received", 0,0);
			LCD.drawInt(length, 0, 1);
			try { Thread.sleep(1000); } catch (InterruptedException e)
{}
	    }catch (IOException ioe) {
			System.err.println("IO Exception writing bytes");
	      }

	    int [][]coords = new int[length][2]; // Δήλωση του πίνακα coords
	    int[] segments = new int[length-1]; // Δήλωση του πίνακα segments
	    int[] angles = new int[length-2]; // Δήλωση του πίνακα angles
	    LCD.clear();
	    LCD.drawString("Done!", 0, 0);
	    LCD.clear();

	    try {
			for(i=0;i<length;i++)
			{
```

```
                        try {
                               LCD.drawString("Receiving...", 0, 0);
                              coords[i][0] = dataIn.readInt(); //
Λαμβάνουμε μία προς μία τις x συντεταγμένες

                              LCD.drawInt(coords[i][0], 0, 1);
                              try { Thread.sleep(100); } catch
(InterruptedException e) {}
                              LCD.clear();

                            } catch (IOException e ) {
                              System.out.println(" write error "+e);
                            }
                    }
                    LCD.drawString("X received", 0, 1);

                    for(i=0;i<length;i++)
                    {
                            try {
                               LCD.drawString("Receiving...", 0, 0);
                              coords[i][1] = dataIn.readInt(); //
Λαμβάνουμε μία προς μία τις y συντεταγμένες

                              LCD.drawInt(coords[i][1], 0, 1);
                              try { Thread.sleep(100); } catch
(InterruptedException e) {}
                              LCD.clear();

                            } catch (IOException e ) {
                              System.out.println(" write error "+e);
                            }
                    }
                    LCD.drawString("Y Received", 0, 2);

                    for(i=0;i<length-1;i++)
                    {
                            try {
                               LCD.drawString("Receiving...", 0, 0);
                              segments[i] = dataIn.readInt(); // Λαμβάνουμε
ένα προς ένα τα μήκη των ευθειών που θα διανύσει το ρομπότ

                              LCD.drawInt(segments[i], 0, 1);
                              try { Thread.sleep(100); } catch
(InterruptedException e) {}
                              LCD.clear();

                            } catch (IOException e ) {
                              System.out.println(" write error "+e);
                            }
                    }
                    LCD.drawString("Segments Received", 0, 3);

                    for(i=0;i<length-2;i++)
                    {
                            try {
```

```
                                LCD.drawString("Receiving...", 0, 0);
                               angles[i] = dataIn.readInt(); // Λαμβάνουμε
μία προς μία τις γωνίες που πρόκειται να στρίψει το ρομπότ

                               LCD.drawInt(angles[i], 0, 1);
                               try { Thread.sleep(100); } catch
(InterruptedException e) {}
                               LCD.clear();

                             } catch (IOException e ) {
                               System.out.println(" write error "+e);
                             }
                    }
                    LCD.drawString("Angles Received", 0, 4);
                  }
            finally
            {try { Thread.sleep(100); } catch (InterruptedException e) {}

            // Εκπέμπουμε 5 φορές έναν ήχο από το ρομπότ για να σηματοδοτήσει
πότε έχει λάβει όλα τα δεδομένα
            for (int k=0;k<5;k++) {
              Sound.beep();
              try { Thread.sleep(500); } catch (InterruptedException e) {}
          }

              // Καλούμε τη μέθοδο pilot της κλάσης Follower, περνώντας ως
ορίσματα τους πίνακες
              // coords, segments, angles και το length
              try {
                           Follower.pilot(coords, segments, angles, length);

                    } catch (Exception e) {
                           e.printStackTrace();
                    }
              LCD.clear();
              LCD.drawString("OK!",0,0);
              connection.close();

        }

        }
}
```

Τέλος Connection.java

DataExchange.java

```
public class DataExchange {
	private int CMD = 0;

	public DataExchange()
	{

	}

	public void setCMD (int command) // H setCMD θέτει στη μεταβλητή CMD το
όρισμα που θα λάβει
	{
		CMD = command;
	}
	public int getCMD() // H getCMD επιστρέφει τη μεταβλητή CMD
	{
		return CMD;
	}

}
```

Τέλος DataExchange.java

Follower.java

```
import lejos.nxt.*;

//motor A είναι ο αισθητήρας απόστασης
//motor B είναι η δεξιά ρόδα κοιτώντας το NXT από πίσω
//motor C είναι η αριστερή ρόδα κοιτώντας το NXT από πίσω
public class Follower {

	public static void pilot(int[][] coords, int[] segments, int[] angles,
int length) throws InterruptedException
	{

		SoundSensor ss = new SoundSensor(SensorPort.S1); // Δημιουργούμε
ένα αντικείμενο ss τύπου SoundSensor που είναι

		 // συνδεδεμένο στη θύρα 1 στο NXT

		LCD.clear(); // Καλούμε αυτή τη μέθοδο για να σβήσουμε ότι είναι
τυπωμένο στην οθόνη

		try { Thread.sleep(1000); } catch (InterruptedException e) {}

		// Όσο το επίπεδο ήχου είναι κάτω από 80 μείνε μέσα στο loop και
		// δείξε μας στην οθόνη πόσο είναι το τρέχον επίπεδο ήχου
		while (ss.readValue()<80)
		{
			LCD.drawInt(ss.readValue(), 0, 0);
		}

		try { Thread.sleep(1000); } catch (InterruptedException e) {}

		DataExchange DE; // Δηλώνουμε ένα αντικείμενο DE τύπου
DataExchange
		IntruderCheck Check; // Δηλώνουμε ένα αντικείμενο Check τύπου
IntruderCheck
		PathFollower PathFollow; // Δηλώνουμε ένα αντικείμενο PathFollow
τύπου PathFollower

		DE = new DataExchange(); //Δημιουργούμε το αντικείμενο
		DE.setCMD(0); // Καλούμε τη μέθοδο setCMD για να δώσουμε τιμή στη
CMD ίση με το 0
		LCD.clear();
		Check = new IntruderCheck(DE); //Δημιουργούμε το αντικείμενο και
δίνουμε το όρισμα
		new Thread(Check).start(); // Καλούμε τη μέθοδο start() για να
ξεκινήσουμε το thread
		PathFollow = new PathFollower(DE, coords.length, segments.length,
angles.length, coords, segments, angles); //Δημιουργούμε το αντικείμενο και
δίνουμε ορίσματα
		new Thread(PathFollow).start(); // Καλούμε τη μέθοδο start() για
να ξεκινήσουμε το thread


		// όσο δεν ακούγεται ήχος μεγαλύτερος του 80 μείνε μέσα στο loop
```

```
            while (ss.readValue()<=80)
            {
                  //LCD.drawInt(DE.getCMD(), 0, 4);
            }

            // Μόλις γίνει ένας δυνατός ήχος βγες από το loop και σταμάτησε
τους κινητήρεσ και σκότωσε τα threads
            Motor.B.flt();
            Motor.C.flt();
            Motor.A.stop();
            Check.kill();
            PathFollow.kill();
      }
```

Τέλος Follower.java

IntruderCheck

```
import lejos.nxt.*;

// Το thread που ελέγχει για εμπόδια
public class IntruderCheck implements Runnable { // Implements Runnable
σημαίνει ότι υπάρχει μέθοδος run() η οποία θα εκτελεί το κομμάτι κώδικα για το
thread

	private UltrasonicSensor uss; // Δήλωση αντικειμένου uss τύπου
UltrasonicSensor
	private DataExchange DEInfo; // Δήλωση αντικειμένου DEInfo τύπου
DataExchange
	boolean killed = false; // Δήλωση και αρχικοποίηση boolean μεταβλητής
killed

	SoundSensor ss = new SoundSensor(SensorPort.S1); // Δημιουργία
αντικειμένου ss τύπου SoundSensor

	public IntruderCheck(DataExchange DE)
	{
		DEInfo = DE; // Δημιουργία αντικειμένου
		uss = new UltrasonicSensor(SensorPort.S4); // Δημιουργία
αντικειμένου
	}
	public void run()
	{

		while(!killed) // όσο η μεταβλητή killed είναι ίση με false τότε
συνέχισε να τρέχει το thread
		{
			uss.ping(); // Μέθοδος για να ελέγξουμε για εμπόδια,
στέλνοντας σήμα υπερήχου από τον αισθητήρα
			if (uss.getDistance()<20) // Αν η απόσταση είναι μικρότερη
από 20
			{
				DEInfo.setCMD(1); // Τότε θέσε τη μεταβλητή CMD από
το αντικείμενο DEInfo ίση με 1
			}
			else
			{
				DEInfo.setCMD(0); // Αλλιώς ίση με 0
			}
		}
	}
	// Μέθοδος με την οποία σκοτώνουμε το thread. Μόλις κληθεί θέτει την
μεταβλητή killed ίση με true
	public void kill()
	{
		killed = true;
	}
}
```

Τέλος IntruderCheck.java

PathFollower.java

```
import java.awt.geom.Line2D;
import lejos.geom.Point;
import lejos.nxt.*;
import lejos.robotics.navigation.DifferentialPilot;
import lejos.robotics.navigation.Pose;
import lejos.robotics.localization.*;


public class PathFollower implements Runnable { // Implements Runnable
σημαίνει ότι υπάρχει μέθοδος run() η οποία θα εκτελεί το κομμάτι κώδικα για το
thread
	// Δήλωση μεταβλητών
	private DataExchange DEInfo; // Αντικείμενο DEInfo τύπου DataExchange
	private int Itinerary = 0; // Μεταβλητή για να ξέρουμε αν πηγαίνει ή
έρχεται στο μονοπάτι
	private int xyLength; // Μέγεθος του πίνακα με τις συντεταγμένες
	private int moveLength; // Μέγεθος του πίνακα με τα μήκη των ευθειών
	private int turnLength; // Μέγεθος πίνακα με τις γωνίες
	private int[][] xy = new int[xyLength][2]; //Δήλωση διδαστατου πίνακα
για να καταχωρήσουμε τις συντεταγμένες x και y
	private int[] move = new int[moveLength]; // Δήλωση μονοδιάστατου πίνακα
για να καταχωρήσουμε τις ευθείες
	private int[] turn = new int[turnLength]; // Δήλωση μονοδιάστατου πίνακα
για να καταχωρήσουμε τις γωνίες
	private int i=0; // Δήλωση μιάς μεταβλητής i
	int distance = 2;

	boolean killed = false; // Δήλωση και αρχικοποίηση boolean μεταβλητής
killed
	SoundSensor ss = new SoundSensor(SensorPort.S1); // Δημιουργία
αντικειμένου ss τύπου SoundSensor
	UltrasonicSensor uss = new UltrasonicSensor(SensorPort.S4); //
Δημιουργία αντικειμένου uss τύπου UltrasonicSensor

	DifferentialPilot robot = new DifferentialPilot(2.1f , 4.5f, Motor.C,
Motor.B); // Δημιουργία αντικειμένου robot τύπου DifferentialPilot δίνοντας
για ορίσματα τη διάμετρο της ρόδας
	// σε ίντσες, την απόσταση των ροδών, τον αριστερό αισθητήρα και τον
δεξί αισθητήρα // Αυτό το αντικείμενο θα κινεί το ρομπότ μας

    OdometryPoseProvider pp = new OdometryPoseProvider(robot); // Δημιουργία
αντικειμένου pp τύπου OdometryPoseProvider περνώντας για όρισμα ένα
DifferentialPilot, και στην προκειμένη το robot
    // Αυτό το αντικείμενο θα είναι υπεύθυνο για την παρακολούθηση της νέας
θέσης του ρομπότ

	Pose pose = new Pose(); // Δημιουργία αντικειμένου pose τύπου Pose //
Αυτό το αντικείμενο θα μας επιτρέψει να περάσουμε για όρισμα στο pp
(OdometryPoseProvider) την τρέχουσα θέση του ρομπότ

	public PathFollower (DataExchange DE, int choordsLength, int
segmentsLength, int angleLength, int[][] coords, int[] segments, int[] angles)
	{
```

```
            // Καταχώρηση μεταβλητών που πήραμε ως ορίσματα
            DEInfo = DE;
            xy = coords;
            move = segments;
            turn = angles;
            xyLength = choordsLength;
            moveLength = segmentsLength;
            turnLength = angleLength;

      }

      // Μέθοδος για να στρίψει το ρομπότ δεξιά
      public void turnLeftWheel()
      {
            robot.rotate(-95);
            sleep(100);
      }

      // Μέθοδος για να στρίψει το ρομπότ αριστερά
      public void turnRightWheel()
      {
            robot.rotate(85);
            sleep(100);
      }

      // Μέθοδος για να προχωρήσει ευθεία
      public void goForward()
      {
            Motor.B.rotate(367,true);
            Motor.C.rotate(367,true);
            sleep(500);
      }

      // Μέθοδος για να κάνει στροφή 360 μοιρών
      public void threeSixty()
      {
            Motor.B.rotate(800);
            sleep(500);
      }

      // Μέθοδος για να κάνουμε τη ζωή μας πιο εύκολη όταν θέλουμε να
"κοιμήσουμε" ένα thread
      // και να μην γράφουμε συνεχώς το {try { Thread.sleep(s); } catch
(InterruptedException e) {}}
      public void sleep(int s)
      {
            {try { Thread.sleep(s); } catch (InterruptedException e) {}}
      }

      // Μέθοδος για να βρει αν το j πρέπει να αυξηθεί ή να μειωθεί αναλόγως
τη φορά του ρομπότ
      public int changeJ(int j)
      {
            if (Itinerary==0)
                  j++;
```

```
            else
                  j--;

            return j;
      }

      // Συνάρτηση υπολογισμού της γωνίας ανάμεσα σε δύο ευθείες που δίνουν ως
ορίσματα
      public static int angleBetween2Lines(Line2D line1, Line2D line2)
    {
            // Υπολογισμός της γωνίας που σχηματίζεται με τις πολικές
συντεταγμένες της μίας ευθείας, με τη χρήση της Math.atan2
            int angle1 = (int) (180/ Math.PI *
(Math.atan2(Math.abs(line1.getY1() - line1.getY2()),
               Math.abs(line1.getX1() - line1.getX2()))));
            // Υπολογισμός της γωνίας που σχηματίζεται με τις πολικές
συντεταγμένες της δεύτερης ευθείας, με τη χρήση της Math.atan2
            int angle2 = (int) (180/ Math.PI *
(Math.atan2(Math.abs(line2.getY1() - line2.getY2()),
               Math.abs(line2.getX1() - line2.getX2()))));
            // Αφαιρώντας τα παραπάνω αποτελέσματα βρίσκουμε τη γωνία ανάμεσα
στις δύο ευθείες
            return angle1-angle2;

    }

      // Η overtake που αφορά την αποφυγή εμποδίου//
      public void overTake()
      {

            int prev; // Μεταβλητή που μας δείχνει το προηγούμενο pixel
            int curr=i; // Μεταβλητή που μας δείχνει το τρέχον pixel
            int next; // Μεταβλητή που μας δείχνει το επόμενο pixel
            int plusFiveAngle=0; // Η γωνία που σχηματίζεται από την ευθεία
του προηγούμενου με το τρέχον pixel και την ευθεία του τρέχοντος με το επόμενο
pixel
            int Cx = 1000;
            int Cy = 1000;
            int Dx = 1000;
            int Dy = 1000;
            int j;
            double minDistance=99999;

            Line2D line1 = new Line2D.Float();
            Line2D line2 = new Line2D.Float();

            if (Itinerary==0)
            {
                  j=i+1;
                  prev = curr - 1;
                  next = curr + 5;
                  if (next>=turnLength)
                  {
                        Itinerary=1;
                        threeSixty();
```

```
                        return;
                }
            }
            else
            {
                j=i-1;
                prev = curr + 1;
                next = curr - 5;
                if (next<=0)
                {
                    Itinerary=1;
                    threeSixty();
                    return;
                }
            }

            line1.setLine(xy[prev][0], xy[prev][1], xy[curr][0],
xy[curr][1]);
            line2.setLine(xy[curr][0], xy[curr][1], xy[next][0],
xy[next][1]);
            plusFiveAngle = angleBetween2Lines(line1,line2);

            pose.setLocation(xy[i][0], xy[i][1]);
            Point nextPoint = new Point(Cx, Cy); // Δημιουργία αντικειμένου
nextPoint τύπου Point
            nextPoint.x=xy[j][0]; // Ορίζουμε τη συντεταγμένη x του nextPoint
            nextPoint.y=xy[j][1]; // Ορίζουμε τη συντεταγμένη y του nextPoint


            pp.setPose(pose); // Μέθοδος με την οποία μας παρέχονται οι
συντεταγμένες και το πού κοιτάζει το ρομπότ

            LCD.clear();
            LCD.drawInt(plusFiveAngle, 0, 0);
            sleep(100);

            LCD.drawInt((int) pose.getX(), 0, 0);
            LCD.drawInt((int) pose.getY(), 5, 0);
            sleep(100);

            // Ο τρόπος που λειτουργεί το παρακάτω κομμάτι κώδικα,
επεξηγείται στο κεφάλαιο 8 του κειμένου της πτυχιακής
            // παρόλα αυτά σε σημεία παρατείθονται σχόλια για την καλύτερη
κατανόηση

            if (plusFiveAngle>=0) // Το μονοπάτι κλίνει προς τα δεξιά
            {
                if (plusFiveAngle<=30) // Το μονοπάτι διέρχεται από την
απέναντι μεριά του εμποδίου
                {
                    turnLeftWheel(); // Στρίβουμε δεξιά
                    sleep(100);

                    Motor.A.rotate(100); // Ο αισθητήρας απόστασης
γυρίζει να κοιτάζει το εμπόδιο
```

```
                        while (uss.getDistance()<40) // Προχωρούμε κάθετα
μέχρις ότου δε βλέπουμε πια το εμπόδιο
                        {
                                robot.forward();
                                uss.ping();
                        }
                        robot.stop();
                        robot.travel(distance);
                        sleep(100);

                        Cx=(int) pp.getPose().getX();
                        Cy=(int) pp.getPose().getY();
                        Point currentPoint = new Point(Cx, Cy); // Βρίσκουμε
το σημείο που είμαστε

                        turnRightWheel(); // Στρίβουμε αριστερά
                        sleep(100);
                        robot.travel(distance);
                        sleep(100);

                        uss.ping();
                        while (uss.getDistance()<40) // Προχωρούμε μέχρις
ότου δε βλέπουμε πια το εμπόδιο
                        {
                                robot.forward();
                                uss.ping();
                        }
                        robot.stop();
                        robot.travel(distance);
                        sleep(100);
                        Motor.A.rotate(-95); // ο αισθητήρας απόστασης
κοίταζει ευθεία

                        // Βρίσκουμε το καινούριο τρέχον σημείο
                        currentPoint.x = (int) pp.getPose().getX();
                        currentPoint.y = (int) pp.getPose().getY();
                        minDistance = currentPoint.distance(nextPoint); //
Αρχικοποιούμε την ελάχιστη απόσταση του τρέχοντος σημείου με το επόμενο

                        LCD.clear();

                        // Αναλόγως προς τα πού κινείται το ρομπότ μπαίνει
στον αντίστοιχο κόμβο του κώδικα
                        if (Itinerary==0)
                        {
                                while(j<xy.length)
                                {
                                        nextPoint.setLocation(xy[j][0],
xy[j][1]);
                                        if
(currentPoint.distance(nextPoint)<minDistance)
                                        {

      minDistance=currentPoint.distance(nextPoint);
```

```
                                        i=j;
                                        j = changeJ(j);
                                        LCD.drawInt(i, 0, 0);
                                }
                                else
                                {
                                        j = changeJ(j);
                                        LCD.drawInt(j, 0, 1);
                                }
                        }
                        j=i+1;
                }
                else
                {
                        while(j>=0)
                        {
                                nextPoint.setLocation(xy[j][0],
xy[j][1]);
                                if
(currentPoint.distance(nextPoint)<minDistance)
                                {

      minDistance=currentPoint.distance(nextPoint);
                                        i=j;
                                        j = changeJ(j);
                                        LCD.drawInt(i, 0, 0);
                                }
                                else
                                {
                                        j = changeJ(j);
                                        LCD.drawInt(j, 0, 1);
                                }
                        }
                        j=i-1;
                }
                // Αφού προσπελάσει όλα τα στοιχεία από το σημείο
που βρέθηκε το εμπόδιο μέχρι τέλος, έχει βρεί το σημείο
                // που απέχει τη μικρότερη απόσταση από το τρέχον
σημείο

                nextPoint.setLocation(xy[i][0], xy[i][1]); // Οπότε
καταχωρούμε τις συντεταγμένες του στο nextPoint

                robot.rotate(currentPoint.angleTo(nextPoint)); //Με
αυτη τη μέθοδο βρίσκουμε προς τα που πρέπει να στρίψει το ρομπότ ώστε να
κοιτάζει το επόμενο σημείο
                sleep(100);
                robot.travel(currentPoint.distance(nextPoint)); //
Με αυτή τη μέθοδο βρίσκουμε πόσο πρέπει να κινηθεί το ρομπότ για να φτάσει σε
αυτό το σημείο
                sleep(100);
                turnLeftWheel();

                pose.setLocation(nextPoint); // Θέτουμε την τρέχουσα
θέση ως το επόμενο σημείο
```

```
                        return;
                }
                // Αντίστοιχα ερμηνεύονται και τα παρακάτω κομμάτια κώδικα
της συνάρτησης
                else if(plusFiveAngle>30 && plusFiveAngle<80)
                {
                        turnLeftWheel();
                        sleep(100);

                        Motor.A.rotate(100);

                        while (uss.getDistance()<40) // Προχωρούμε κάθετα
μέχρις ότου δε βλέπουμε πια το εμπόδιο
                        {
                                robot.forward();
                                uss.ping();
                        }
                        robot.stop();
                        robot.travel(4);
                        sleep(100);

                        Cx=(int) pp.getPose().getX();
                        Cy=(int) pp.getPose().getY();
                        Point currentPoint = new Point(Cx, Cy);

                        turnRightWheel();

                        Motor.A.rotate(-90);
                        LCD.clear();
                        if (Itinerary==0)
                        {
                                while(j<xy.length)
                                {
                                        nextPoint.setLocation(xy[j][0],
xy[j][1]);
                                        if
(currentPoint.distance(nextPoint)<minDistance)
                                        {

      minDistance=currentPoint.distance(nextPoint);
                                                i=j;
                                                j = changeJ(j);
                                                LCD.drawInt(i, 0, 0);
                                        }
                                        else
                                        {
                                                j = changeJ(j);
                                                LCD.drawInt(j, 0, 1);
                                        }
                                }
                                j=i+1;
                        }
                        else
                        {
                                while(j>=0)
```

```
                                {
                                        nextPoint.setLocation(xy[j][0],
xy[j][1]);
                                        if
(currentPoint.distance(nextPoint)<minDistance)
                                        {

      minDistance=currentPoint.distance(nextPoint);
                                                i=j;
                                                j = changeJ(j);
                                                LCD.drawInt(i, 0, 0);
                                        }
                                        else
                                        {
                                                j = changeJ(j);
                                                LCD.drawInt(j, 0, 1);
                                        }
                                }
                                j=i-1;
                        }
                        nextPoint.setLocation(xy[i][0], xy[i][1]);

                        robot.rotate(currentPoint.angleTo(nextPoint));
                        sleep(100);
                        robot.travel(currentPoint.distance(nextPoint));
                        sleep(100);
                        turnLeftWheel();

                        pose.setLocation(nextPoint);
                        return;

                }
                else if (plusFiveAngle>=80)
                {
                        Cx=(int) pp.getPose().getX();
                        Cy=(int) pp.getPose().getY();
                        Point currentPoint = new Point(Cx, Cy);
                        nextPoint.setLocation(xy[next][0],xy[next][1]);

                        robot.rotate(currentPoint.angleTo(nextPoint));
                        sleep(100);
                        robot.travel(currentPoint.distance(nextPoint));
                        sleep(100);
                        turnLeftWheel();

                        pose.setLocation(nextPoint);
                        return;
                }
        }
        else
        {
                if (plusFiveAngle>=-30)
                {
                        turnRightWheel();
                        sleep(100);
```

```
                        Motor.A.rotate(-90);

                        while (uss.getDistance()<40) // Προχωρούμε κάθετα
μέχρις ότου δε βλέπουμε πια το εμπόδιο
                        {
                                robot.forward();
                                uss.ping();
                        }
                        robot.stop();
                        robot.travel(4);
                        sleep(100);

                        Cx=(int) pp.getPose().getX();
                        Cy=(int) pp.getPose().getY();
                        Point currentPoint = new Point(Cx, Cy);

                        turnLeftWheel();
                        robot.travel(4);
                        sleep(100);

                        uss.ping();
                        while (uss.getDistance()<40)
                        {
                                robot.forward();
                                uss.ping();
                        }
                        sleep(100);
                        robot.stop();
                        sleep(100);
                        Motor.A.rotate(100);

                        currentPoint.x = (int) pp.getPose().getX();
                        currentPoint.y = (int) pp.getPose().getY();
                        minDistance = currentPoint.distance(nextPoint);

                        LCD.clear();

                        if (Itinerary==0)
                        {
                                while(j<xy.length)
                                {
                                        nextPoint.setLocation(xy[j][0],
xy[j][1]);
                                        if
(currentPoint.distance(nextPoint)<minDistance)
                                        {

      minDistance=currentPoint.distance(nextPoint);
                                                i=j;
                                                j = changeJ(j);
                                                LCD.drawInt(i, 0, 0);
                                        }
                                        else
                                        {
```

```
                                        j = changeJ(j);
                                        LCD.drawInt(j, 0, 1);
                                }
                        }
                        j=i+1;
                }
                else
                {
                        while(j>=0)
                        {
                                nextPoint.setLocation(xy[j][0],
xy[j][1]);
                                if
(currentPoint.distance(nextPoint)<minDistance)
                                {

      minDistance=currentPoint.distance(nextPoint);
                                        i=j-1;
                                        j = changeJ(j);
                                        LCD.drawInt(i, 0, 0);
                                }
                                else
                                {
                                        j = changeJ(j);
                                        LCD.drawInt(j, 0, 1);
                                }
                        }
                        j=i-1;
                }
                nextPoint.setLocation(xy[i][0], xy[i][1]);

                robot.rotate(currentPoint.angleTo(nextPoint));
                sleep(100);
                robot.travel(currentPoint.distance(nextPoint));
                sleep(100);
                turnRightWheel();

                pose.setLocation(nextPoint);
                return;
        }
        else if (plusFiveAngle<-30 && plusFiveAngle>-80)
        {
                turnRightWheel();
                sleep(100);

                Motor.A.rotate(-90);


                while (uss.getDistance()<40) // Προχωρούμε κάθετα
μέχρις ότου δε βλέπουμε πια το εμπόδιο
                {
                        robot.forward();
                        uss.ping();
                }
                robot.stop();
```

```
                        robot.travel(4);
                        sleep(100);

                        Cx=(int) pp.getPose().getX();
                        Cy=(int) pp.getPose().getY();
                        Point currentPoint = new Point(Cx, Cy);

                        turnLeftWheel();

                        Motor.A.rotate(100);
                        LCD.clear();
                        if (Itinerary==0)
                        {
                               while(j<xy.length)
                               {
                                      nextPoint.setLocation(xy[j][0],
xy[j][1]);
                                      if
(currentPoint.distance(nextPoint)<minDistance)
                                      {

      minDistance=currentPoint.distance(nextPoint);
                                             i=j;
                                             j = changeJ(j);
                                             LCD.drawInt(i, 0, 0);
                                      }
                                      else
                                      {
                                             j = changeJ(j);
                                             LCD.drawInt(j, 0, 1);
                                      }
                               }
                               j=i+1;
                        }
                        else
                        {
                               while(j>=0)
                               {
                                      nextPoint.setLocation(xy[j][0],
xy[j][1]);
                                      if
(currentPoint.distance(nextPoint)<minDistance)
                                      {

      minDistance=currentPoint.distance(nextPoint);
                                             i=j;
                                             j = changeJ(j);
                                             LCD.drawInt(i, 0, 0);
                                      }
                                      else
                                      {
                                             j = changeJ(j);
                                             LCD.drawInt(j, 0, 1);
                                      }
                               }
```

```
                                j=i-1;
                        }
                        nextPoint.setLocation(xy[i][0], xy[i][1]);

                        robot.rotate(currentPoint.angleTo(nextPoint));
                        sleep(100);
                        robot.travel(currentPoint.distance(nextPoint));
                        sleep(100);
                        turnRightWheel();

                        pose.setLocation(nextPoint);
                        return;
                }
                else if (plusFiveAngle<=-80)
                {
                        Cx=(int) pp.getPose().getX();
                        Cy=(int) pp.getPose().getY();
                        Point currentPoint = new Point(Cx, Cy);
                        nextPoint.setLocation(xy[next][0],xy[next][1]);

                        robot.rotate(currentPoint.angleTo(nextPoint));
                        sleep(100);
                        robot.travel(currentPoint.distance(nextPoint));
                        sleep(100);
                        turnRightWheel();

                        pose.setLocation(nextPoint);
                        return;
                }

        }
    }

    public void run()
    {
        i=0;
        LCD.clear();
        pose.setLocation(xy[0][0], xy[0][1]); // Θέτουμε ως αρχική θέση
το πρώτο pixel
        robot.setRotateSpeed(80); // Μέθοδος με την οποία ορίζουμε την
ταχύτητα που θα στρίβει το ρομπότ
        robot.setTravelSpeed(10); // Μέθοδος με την οποία ορίζουμε την
ταχύτητα που θα διανύει μία ευθέια
        while (!killed)  // όσο η μεταβλητή killed είναι ίση με false
τότε συνέχισε να τρέχει το thread
        {
                if(DEInfo.getCMD()==0) // Αν δεν έχει εντοπίσει εμπόδιο
στο μονοπάτι
                {

                        if (Itinerary==0) // Αν το ρομπότ πηγαίνει από το
πρώτο pixel στο τελευταίο
                        {
                                //Motor.B.rotate(move[i], true);
                                //Motor.C.rotate(move[i], true);
```

```
                                robot.travel(distance, true); // Προχώρησε
κατά distance
                                LCD.drawInt(move[i],0,0);
                                LCD.drawInt(i, 0, 1);
                                LCD.drawInt(turnLength, 0, 2);
                                sleep(100); // Σταμάτα για ένα δέκατο του
δευτερολέπτου
                                if (i>turnLength-1) // Αν ο μετρητής είναι
μεγαλύτερος από turnLength-1 σημαίνει οτι δεν υπάρχει άλλο στοιχείο στον
πίνακα με τις γωνίες
                                {
                                      i=moveLength-1; // Επομένως θέτουμε στο
i την τιμή moveLength-1
                                      Itinerary=1; // και εφόσον φτάσαμε στο
τέλος του μονοπατιού η Itinerary γίνεται 1 και σηματοδοτεί πως το ρομπότ
επιστρέφει
                                                                  // δηλαδή
προσπελαύνει τα στοιχεία από το τελευταίο προς το πρώτο
                                      threeSixty(); // Κανε στροφή 360 μοιρών
                                      LCD.clear();
                                      Sound.beep(); // Κάνε ένα ήχο
                                }
                                else // αλλιώς αν υπάρχει κι άλλο στοιχείο
στο πίνακα με τις γωνίες, τότε διάβασέ το
                                {
                                      if (turn[i]>0) // αν είναι μεγαλύτερο
του μηδέν τότε πρέπει να στρίψει αριστερά, άρα περιστρέφουμε τον δεξί τροχό
                                      {
                                            Motor.B.rotate(turn[i]);

                                      }
                                      else if(turn[i]<0) // αλλιώς
περιστρέφουμε τον αριστερό τροχό
                                      {

      Motor.C.rotate(Math.abs(turn[i]));
                                      }

                                      LCD.drawInt(turn[i],5,0);
                                      LCD.drawInt(i, 5, 1);
                                      LCD.drawInt(Itinerary, 5, 2);
                                      i++;
                                }

                          }
                          else // Η Itinerary είναι ίση με το 1 άρα το ρομπότ
περνά τα στοιχεία από το τελευταίο προς το πρώτο
                          {
                                robot.travel(distance, true);
                                sleep(100);
                                i--;
                                if(i<0)
                                {
                                      i=0;
                                      Itinerary=0;
```

```
                                        threeSixty();
                                        LCD.clear();
                                        Sound.beep();
                                }
                                else
                                {
                                        if (turn[i]>0) // Εδώ επειδή
επιστρέφουμε, οι ρόλοι των τροχών θα αντιστραφούν
                                                       // Με άλλα λόγια όταν η γωνία
είναι μεγαλύτερη του μηδέν θα περιστρέφουμε τον αριστερό τροχό
                                        {
                                                Motor.C.rotate(turn[i]);
                                        }
                                        else if(turn[i]<0) // Αλλιώς
περιστρέφουμε το δεξί τροχό
                                        {

      Motor.B.rotate(Math.abs(turn[i]));
                                        }
                                        LCD.drawInt(turn[i],5,0);
                                        LCD.drawInt(i, 5, 1);
                                        LCD.drawInt(Itinerary, 5, 2);
                                }

                        }
                }
                else // Αν η CMD του DEInfo είναι ίση με 1 τότε έχει
εντοπίσει εμπόδιο και επομένως
                {
                        pose.setLocation(xy[i][0], xy[i][1]); // θέτουμε τον
τρέχον σημείο ως την τρέχουσα θέση
                        overTake(); // και καλούμε την overTake()

                }
        }
      }

      // Μέθοδος με την οποία σκοτώνουμε το thread. Μόλις κληθεί θέτει την
μεταβλητή killed ίση με true
      public void kill()
      {
            killed = true;
      }
}
```

Τέλος PathFollower.java

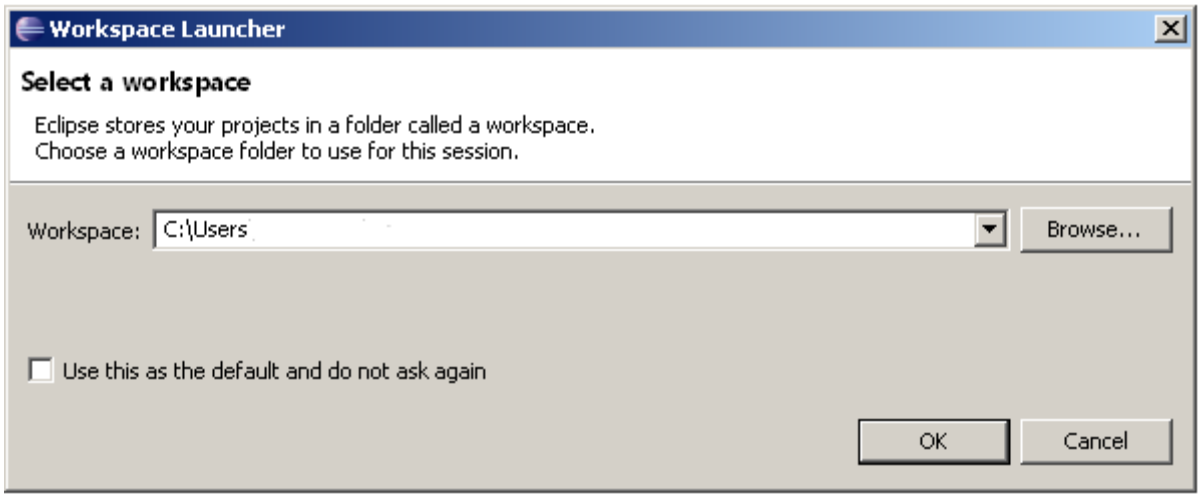
ΠΑΡΑΡΤΗΜΑ Β

Παρατίθεται βήμα βήμα η εγκατάσταση των οδηγών και των προγραμμάτων που ήταν απαραίτητα για αυτή την εργασία.

Eclipse

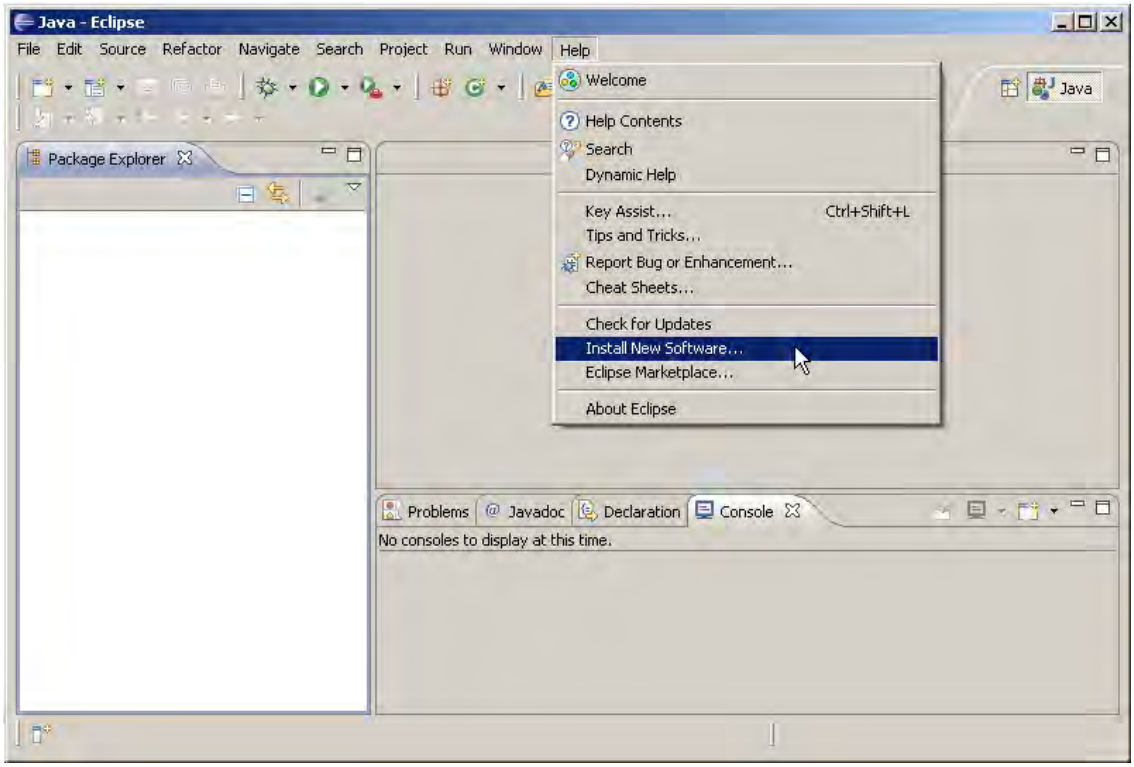
Μπορείτε να λάβετε το Eclipse από αυτή τη σελίδα http://www.eclipse.org/downloads/

Μόλις το κατεβάσετε μπορείτε να τρέξετε την εγκατάσταση του. Όταν ολοκληρωθεί η εγκατάσταση, τρέξτε το και θα σας ζητηθεί να ορίσετε το workspace (ο φάκελος όπου θα αποθηκεύονται τα προγράμματα που θα γράψετε) όπως φαίνεται στην παρακάτω εικόνα:



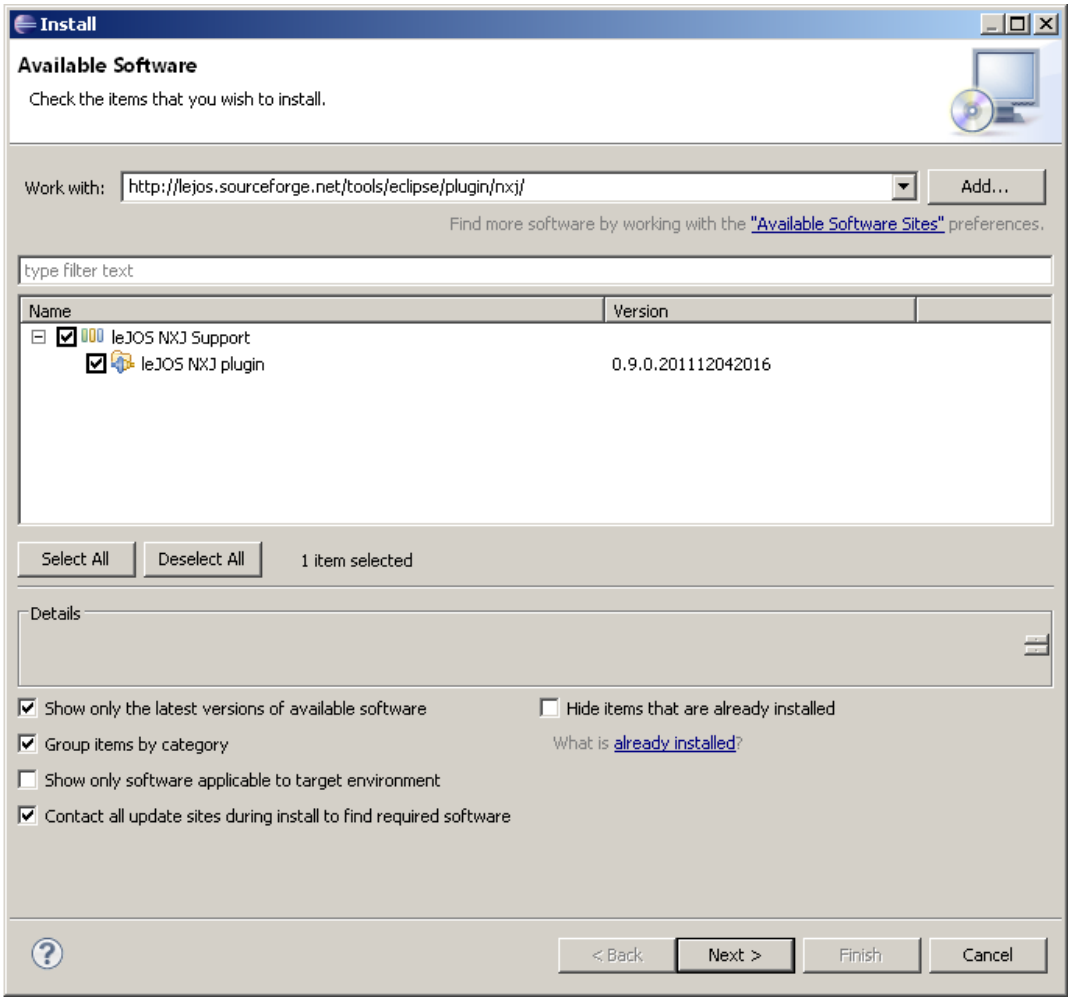
Eclipse Plugin

Το plugin για το Eclipse, θα σας επιτρέψει να δημιουργήσετε προγράμματα NXT για τον υπολογιστή και το ρομπότ Lego Mindstorms NXT, με τη δυνατότητα να τα μεταγλωτίσετε και να τα τρέξετε με το πάτημα ενός πλήκτρου και να σταλούν μέσω USB ή Bluetooth. Για να το εγκαταστήσετε, πηγαίνετε στο κεντρικό παράθυρο του Eclipse, πατάτε “Help” και μετά “Install New Software…” όπως φαίνεται στην παρακάτω εικόνα:



Στη συνέχεια πληκτρολογείτε το εξής στο πεδίο “Work with:”

http://www.lejos.org/tools/eclipse/plugin/nxj/ και πατάτε στο πληκτρολόγιο το “Enter”



Διαλέξτε το leJOS plugin και πατήστε “Next” στις επόμενες σελίδες και τελικά πατήστε στο “Finish”

Όταν σας ζητηθεί επανεκινήστε στο Eclipse.

Εγκατάσταση οδηγών και Firmware

Java Development Kit

Θα χρειατείτε ένα Java Development Kit (JDK) στον υπολογιστή σας. Μπορείτε να το κατεβάσετε την πιο τελευταία έκδοση από εδώ:
http://www.oracle.com/technetwork/java/javase/downloads/

Οδηγός USB

Για να υπάρξει ένας τρόπος επικοινωνίας ανάμεσα στον υπολογιστή και στο ρομπότ θα πρέπει να εγκαταστήσετε τον οδηγό Fantom για το USB. Μπορείτε να τον βρείτε εδώ:
http://www.lego.com/en-us/mindstorms/downloads/software/ddfiledownload/

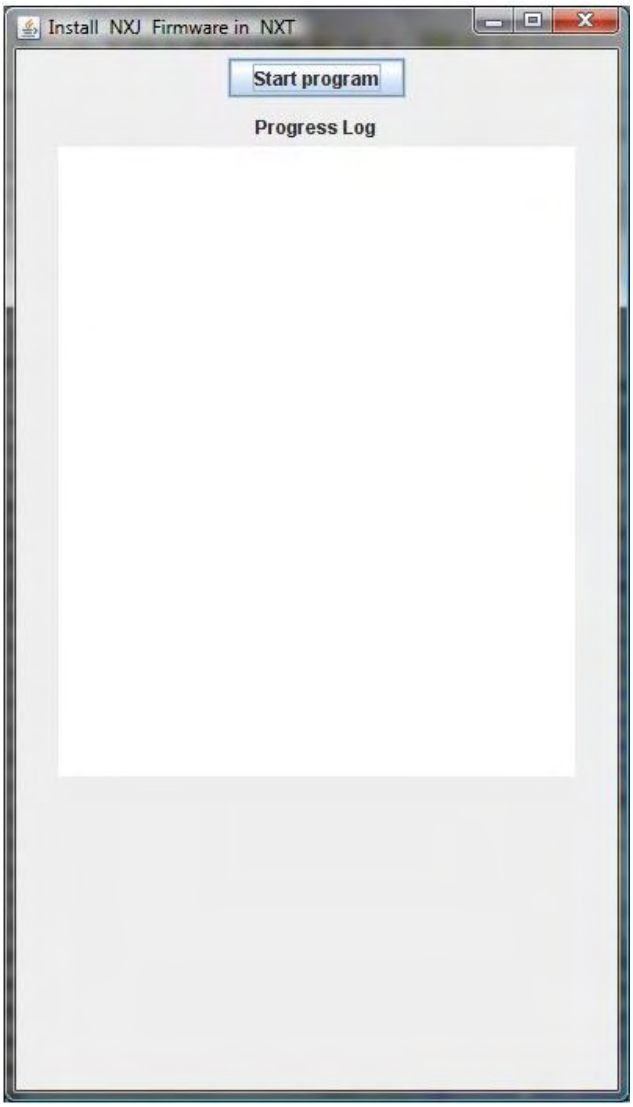
leJOS Firmware

Μπορείτε να το κατεβάσετε από εδώ: http://www.lejos.org/nxj-downloads.php

Κατεβάστε το **leJOS_NXJ_0.9.0-Setup.exe** και τρέξτε το.

Πατήστε "Next" και όταν σας ζητηθεί διαλέξτε το φάκελο που εγκαταστήσατε το Java Development Kit (JDK). Πατήστε "Next" στις επόμενες σελίδες. Όταν τελειώσει η εγκατάσταση θα εμφανιστεί το παρακάτω:

Αν το αφήσετε τσεκαρισμένο, σημαίνει ότι θέλετε να εγκαταστήσετε το λειτουργικό σύστημα leJOS στο ρομπότ NXT. Πατώντας “Finish” θα εμφανιστεί το παρακάτω:



Μόλις πατήσετε “Start Program” θα εμφανιστεί ένα παράθυρο που θα σας ζητήσει να πατήσετε “OK” όταν το ρομπότ NXT είναι συνδεδεμένο με USB και αναμένο. Πατώντας “OK” θα ερωτηθείτε αν θέλετε να σβήσετε τα αρχεία του NXT. Πατήστε ναι, και θα σας ξαναζητηθεί να πατήσετε “OK” όταν το ρομπότ NXT είναι συνδεδεμένο και αναμένο. Πατήστε “OK” και μετά από λίγη ώρα η εγκατάσταση θα τελειώσει και θα εμφανιστεί ένα παράθυρο που θα σας ρωτήσει αν θέλετε να επαναλάβετε τη διαδικασία. Πατήστε όχι και είστε έτοιμοι.