ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΤΜΗΜΑ
ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΪΑΤΡΙΚΗ

Ευφυής έλεγχος και πλοήγηση ρομπότ μέσω της πλατφόρμας ARIA

Advanced Robot Interface for Applications

Γιαχούδης Νικόλαος

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ
Επιβλέπων
Πλαγιανάκος Βασίλης
Επίκουρος Καθηγητής

Λαμία, 2014

Περιεχόμενα

Περίληψη

Ο κόσμος αναπτύσσεται τεχνολογικά με ραγδαίους ρυθμούς. Συνέχεια αναπτύσσονται εφαρμογές τόσο βιομηχανικές όσο και ιατρικές, αλλά και εφαρμογές που έχουν στόχο την διασκέδαση. Πολλές από τις εφαρμογές έχουν να κάνουν με κάποια ρομποτικά συστήματα ή τουλάχιστον με κάποια μορφή τεχνητής νοημοσύνης. Θεωρείται ιδιαίτερα σημαντική η συνεχής έρευνα πάνω σε αυτούς τους τομείς. Με τη λογική αυτή πραγματοποιήσαμε μια πτυχιακή πάνω σε τομείς της ρομποτικής και της τεχνητής νοημοσύνης.

Με την πτυχιακή εργασία προσπαθήσαμε να φτιάξουμε ένα σύστημα, το οποίο θα μπορεί να χειριστεί ένα ρομπότ αποδοτικά. Δηλαδή θα πρέπει να αποφεύγει εμπόδια σε πραγματικό χρόνο και αχαρτογράφητα. Θα πρέπει να γνωρίζει την θέση του στο χώρο, και οι βασικές του λειτουργίες θα είναι να ακολουθεί και να ψάχνει έναν συγκεκριμένο στόχο.

Abstract

The world progresses rapidly in all technological sciences. New applications for industrial use and medical use are developed. The most of them are designed using some form of robotics or artificial intelligence. So it must be important to increase our research effort on this scientific field. Having this in mind we focused on those domains to study about.

In this report we have tried to implement a system that can navigate a robot efficiently and effectively. The robot must bypass any obstacles in its way in real time and even obstacles that are dynamically moving in the environment. The robot must know and be able to find its position in space. And its basic functions will be to searching and follow a particular target.

Ευχαριστίες

Ευχαριστώ τους γονείς μου, που μου συμπαραστάθηκαν σε αυτή την προσπάθεια μου. Την αδερφή μου γιατί ξέρω ότι πάντα πίστευε σε μένα και θα συνεχίσει έτσι. Φυσικά θα ήθελα να ευχαριστήσω τον κύριο Πλαγιανάκο γιατί μου έδωσε την ευκαιρία να ασχοληθώ με κάτι που πραγματικά επιθυμούσα, την κυρία Αδάμ, τον κύριο Μάρκου αλλά και όλους τους καθηγητές μου που μου χάρισαν όλη αυτή την υπέροχη γνώση. Ένα μεγάλο ευχαριστώ σε όλους τους φίλους μου, γιατί αυτοί ήταν που με παρότρυναν πιο πολύ από όλους. Αλλά δεν θα κατάφερνα τίποτα, ούτε θα είχα στόχους, αν δεν υπήρχε μια συγκεκριμένη κοπέλα στη ζωή μου. Την ευχαριστώ και της αφιερώνω αυτή την εργασία.

Κεφάλαιο 1

Εισαγωγή

1.1 Τεχνητή νοημοσύνη

Ο όρος Τεχνητή Νοημοσύνη – ΤΝ(Artificial Intelligence – AI) υπάρχει εδώ και 50 χρόνια και όμως παραμένει από τις ποιο επίκαιρες επιστήμες του κλάδου της επιστήμης των υπολογιστών. Παρόλα αυτά η εξερεύνηση της νοημοσύνης γενικότερα ήταν θέμα για συζήτηση από την εποχή μεγάλων φιλοσόφων όπως ο Αριστοτέλης, ο Ηράκλειτος, ο Descartes. Επίσης η αναζήτηση για την κατανόηση του μηχανισμού της μάθησης, της απομνημόνευσης, της όρασης, της αντίληψης και του συλλογισμού. Η έρευνα στην τεχνητή νοημοσύνη έχει αρχίσει να ανθίζει όχι πολλα χρόνια πριν. Η πρωταρχική συζήτηση για την ΤΝ έλαβε μέρος το 1956 σε μια συνάντηση των John McCarthy, Marvin Minsky, Claude Shannon.

Η ΤΝ έχει μεγάλο εύρος ερευνητικού πεδίου που εμπεριέχει γενικές κατηγορίες όπως η αντίληψη και η συλλογιστική, αλλά και πιο συγκεκριμένα πεδία όπως η θεωρία παιγνίων, η απόδειξη θεωρημάτων, η διάγνωση ασθενειών. Γενικά οι ερευνητές καταφεύγουν στην ΤΝ για να βρουν κάποια εργαλεία που θα τους φανούν χρήσιμα στην έρευνά τους.

1.1.1 Τι είναι η τεχνητή νοημοσύνη

Υπάρχουν πολλοί και διαφορετικοί ορισμοί για την ΤΝ. Κάποιοι από αυτούς είναι:

- «Η ικανότητα ενός Η/Υ ή ρομπότ που ελέγχεται από ένα Η/Υ, να πραγματοποιεί συμπεριφορές που συνήθως συνδέονται με νοήμονα όντα.»
- «Με τον όρο *Τεχνητή Νοημοσύνη* εννοούμε την χρήση προγραμμάτων και προγραμματιστικών τεχνικών, για να κατανοήσουμε γενικότερα τις αρχές της νοημοσύνης και της ανθρώπινης σκέψης πιο συγκεκριμένα.»

Άλλοι ορισμοί είναι:

- «Η προσπάθεια να κατασκευάσουμε υπολογιστές με διανοητική ικανότητα με την πλήρη και κυριολεκτική έννοια του όρου» σύμφωνα με τον Haugeland.

- «Η μελέτη των υπολογισμών που καθιστούν εφικτή την αντίληψη, τη λογική σκέψη και την ανάδραση» όπως λέει ο Winston.

- Οι Rich και Knight ορίζουν την ΤΝ ως «Τη μελέτη του πώς κάνουμε τους υπολογιστές να κάνουν πράγματα στα οποία αυτήν τη στιγμή ο άνθρωπος είναι καλύτερος.»

- Και ο Luger λέει ότι η ΤΝ είναι «Ο τομέας της επιστήμης των υπολογιστών που ασχολείται με την αυτοματοποίηση της ευφυούς συμπεριφοράς.»

Παρατηρούμε λοιπόν ότι η ΤΝ ορίζεται με πολλούς τρόπους, γιατί υπάρχουν και πολλές οπτικές γωνίες, από τις οποίες μπορεί να γίνει αντιληπτή. Για παράδειγμα από τη σκοπιά της ανάπτυξης συστημάτων που σκέφτονται όπως ο άνθρωπος ή από τη σκοπιά των συστημάτων που δρουν λογικά. Γίνεται κατανοητό λοιπόν, ότι ο κλάδος της επιστήμης των υπολογιστών που ονομάζεται τεχνητή νοημοσύνη, εξελίσσεται και αναπτύσσεται συνεχώς και έχει αρχίσει να γίνεται ένα ξεχωριστό επιστημονικό πεδίο από μόνο του.

Σε βασικές γραμμές όμως η κύρια προσέγγιση της τεχνητής νοημοσύνης βασίζεται σε σύμβολα. Έτσι, με ένα σύνολο συμβόλων μπορούμε χωρίς πολλά λόγια να εκφράσουμε τόσο ένα πρόβλημα όσο και την επίλυση του. Είναι ένας τρόπος δηλαδή για την εύκολη και γρήγορη κατανόηση ενός προβλήματος και πιθανότατα της λύσης του. Δεν θα μπορούσαμε όμως μόνο με σύμβολα να αποσαφηνίσουμε πλήρως μια συγκεκριμένη όψη του κόσμου. Έτσι, χρησιμοποιούμε μια πιο αφηρημένη όψη του κόσμου μέσω συμβόλων τα οποία τα επεξεργαζόμαστε.

Όμως η λειτουργία του ανθρώπινου εγκεφάλου δεν λειτουργεί κατ' αυτήν την έννοια. Ο ανθρώπινος εγκέφαλος δεν αποθηκεύει σε συγκεκριμένα σημεία αντίστοιχα μνήμες, σκέψεις, λειτουργίες. Αλλά σε οτιδήποτε έχει να κάνει με νοητική διεργασία χρησιμοποιείται το μεγαλύτερο μέρος του εγκεφάλου ταυτόχρονα. Αυτό βέβαια φαίνεται στο ότι αν πάθει κάποια βλάβη συγκεκριμένο τμήμα του, δεν συγκεκριμενοποιείται και η βλάβη.

Έτσι έχουμε δυο προσεγγίσεις για την ΤΝ:

- Η κλασική ή συμβολική τεχνητή νοημοσύνη, που προσεγγίζει την ανθρώπινη νοημοσύνη με αλγόριθμους και σύμβολα.

- Η υπολογιστική τεχνητή νοημοσύνη, που μιμείται σχεδόν κυριολεκτικά βιολογικές διεργασίες. Όπως τα τεχνητά νευρωνικά δίκτυα και οι γενετικοί αλγόριθμοι.

Φυσικά δε θα μπορούσαμε να αφήσουμε εκτός συζήτησης τον Alan Turing (1912-1954). Ο Turing, ο οποίος θεωρείται από πολλούς ο πατέρας της ΤΝ, σκέφτηκε μια διαδικασία ή μπορούμε να πούμε αλλιώς ένα τεστ, που αν το εφαρμόσεις μπορείς να πεις με βεβαιότητα αν κάποια συμπεριφορά πηγάζει από την νοημοσύνη.

Η δοκιμασία αυτή έχει ως εξής:

Ο εξεταστής, ο οποίος είναι άνθρωπος, κάνει ερωτήσεις σε μια μηχανή και έναν άνθρωπο ταυτόχρονα. Χωρίς όμως να γνωρίζει ποιος είναι ο καθένας. Αν λοιπόν από της απαντήσεις που λαμβάνει δεν μπορεί να ξεχωρίσει το τεχνητό

από τον άνθρωπο, τότε η μηχανή περνάει επιτυχώς την δοκιμασία και μπορεί πλέον να θεωρείται ευφυής.

Για να επιτευχθεί αυτό πρέπει να αναλογιστούμε πόσα διαφορετικά επιστημονικά πεδία παίρνουν μέρος. Πρέπει να υπάρχει επεξεργασία της φυσικής γλώσσας (natural language processing) για την επικοινωνία, αναπαράσταση γνώσης (knowledge representation) για την αποθήκευση της πριν και κατά τη διάρκεια της δοκιμασίας, μηχανική μάθηση (machine learning) για προσαρμογή σε καινούριες πιθανές ερωτήσεις και πολλά άλλα.

1.1.2 Εξέλιξη της τεχνητής νοημοσύνης

Το επιστημονικό πεδίο της ΤΝ αν και είναι σχετικά νέο, κληρονομεί πολλά στοιχεία και τεχνικές από άλλες επιστήμες όπως η γλωσσολογία, η ψυχολογία και τα μαθηματικά. Την πρώτη ουσιαστική πρόοδο που είχε όμως την οφείλουμε στον George Boole (1854). Ο Boole είναι αυτός που θεμελίωσε την έναρξη της ΤΝ θέτοντας τις βάσεις της προτασιακής λογικής. Πρότεινε το δυαδικό σύστημα μαζί με ένα σύνολο συμβόλων για όλες της λογικές πράξεις που μπορούμε να έχουμε ανάμεσα στο 1 και στο 0.

Έτος	**Ορόσημο**
1833	Ο Charles Babbage, που θεωρείται από πολλούς και ο πατέρας των υπολογιστών, πρότεινε την *Αναλυτική Μηχανή*, μια υπολογιστική μηχανή που εκτελούσε εντολές.
1842	Η Ada Byron, κόρη του ποιητή λόρδου Βύρωνα, έγραψε τα πρώτα προγράμματα για τη μηχανή του Babbage.
1854	Ο George Boole, δημιούργησε την άλγεβρα Boole.
1889	Ο Herman Hollerith, πατέρας της στατιστικής, εφηύρε τις διάτρητες κάρτες που χρησιμοποιήθηκαν στην απογραφή πληθυσμού. Η εταιρεία που ίδρυσε ονομάστηκε αργότερα IBM.
1936	Ο Alan Turing πρότεινε μια μηχανή (Turing Machine) που είχε τη δυνατότητα να εκτελεί οποιαδήποτε λειτουργία και η οποία μπορούσε να περιγραφεί με μια πεπερασμένη ακολουθία οδηγιών (εντολών). Ουσιαστικά έθεσε τις βάσεις για τους σύγχρονους υπολογιστές.
1938	Ο Konrad Zuse κατασκεύασε στην Γερμανία τον Z1, τον πρώτο ηλεκτρομηχανικό προγραμματιζόμενο υπολογιστή.
1945	Ο John Von Newman, σχεδίασε τον πρώτο υπολογιστή EDVAC (γνωστός και ως μηχανή Von Newman) με μνήμη, επεξεργαστή και μονάδες εισόδου/εξόδου. Αποτέλεσε τη βάση του ENIAC.
1946	Κατασκευάστηκε στην Πενσιλβανία, ο ENIAC, ο πρώτος ηλεκτρονικός υπολογιστής (η κατασκευή ξεκίνησε το 1943).
1948	Ο Claude Shannon δημοσίευσε το "Mathematical Theory of Communication", θεμελιώνοντας την θεωρία πληροφοριών στην οποία στηρίχθηκαν οι ψηφιακές επικοινωνίες.

Πίνακας 1.1: Ορόσημα στην εξέλιξη των υπολογιστών.

Σχήμα 1.1: Deep Blue το ευφυές σύστημα που νίκησε τον σκακιστή Garry Kasparov

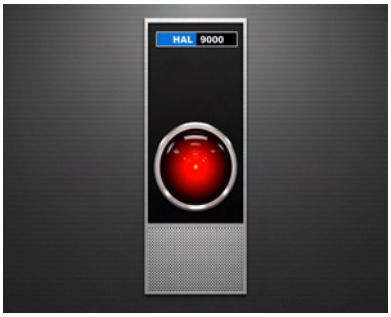
Όπως είδαμε και παραπάνω σημαντική ήταν η συνάντηση των John McCarthy, Marvin Minksy, Claude Shannon και Nathaniel Rochester στο Dartmouth της Μασαχουσέτης το 1956. Εκεί συγκεντρώθηκαν για να μιλήσουν περί θεωρίας αυτομάτων, νευρωνικών δικτύων και μελέτης ευφυίας. Δυο ερευνητές ο Allen Newell και Herbert Simon παρουσίασαν το πρόγραμμα LOGIC THEORIST, το οποίο ήταν ένα πρόγραμμα συλλογισμού. Αυτό μπορούσε να αποδείξει πολλά από τα θεωρήματα των Russell και Whitehead μέσα από τα Principia Mathematica. Τελικά στο τέλος της συζήτησης ο McCarthy πρότεινε το όνομα τεχνητή νοημοσύνη για τη νέα αυτή ερευνητική περιοχή.

Τη δεκαετία του '60 αναπτύχθηκαν πολύ τα νευρωνικά, με τη βελτίωση του αλγορίθμου μάθησης του Hebb από τον Rosenblatt (1962). Ο Rosenblatt λοιπόν, απέδειξε τη σύγκλιση των perceptron, δηλαδή έδειξε ότι προσαρμόζονται τα βάρη στους νευρώνες ενός ΤΝΔ έτσι ώστε η έξοδος να συγκλίνει στην επιθυμητή έξοδο, αρκεί αυτή να υπάρχει.

Δυστυχώς όμως μετά την πρώτη δεκαετία του ενθουσιασμού ήρθε μεγάλη και βαριά κριτική στην περιοχή της ΤΝ. Αυτό που υποστηριζόταν ήταν ότι τα συστήματα αυτά είναι ικανά να λύσουν μόνο μικροπροβλήματα και ότι δεν θα μπορέσουν να αντεπεξέλθουν στην αύξηση της δυσκολίας των προβλημάτων. Έτσι το 1973 η Βρετανική κυβέρνηση διέκοψε την χρηματοδότηση της έρευνας στην ΤΝ.

Επίσης την δεκαετία του '80 επανεμφανίστηκαν τα νευρωνικά δίκτυα (Hopfield, 1982) όπως και ο αλγόριθμος μάθησης με ανάστροφη μετάδοση του λάθους (back-propagation) που προτάθηκε από τους Bryson και Ho. Εφαρμόστηκε με επιτυχία σε πολλά προβλήματα.

Έτσι μπορούμε να δούμε ότι αν και με μόνο 50 χρόνια ζωής η ΤΝ έχει περάσει διάφορα στάδια «ζωής». Εκτός όμως από το επιστημονικό κομμάτι η ΤΝ είχε πρωτοεμφανισθεί σε πολλά διηγήματα επιστημονικής φαντασίας. Γνωστοί συγγραφείς της ΕΦ όπως ο Isaac Asimov και ο Arthur C. Clarke έχουν συμβάλει και οι ίδιοι κατά κάποιο τρόπο στην εξέλιξη της τεχνητής νοημοσύνης.



Σχήμα 1.2: Hal 9000 από την ταινία Space Odyssey του Cubrik βασισμένη στο ομώνυμο βιβλίο του Arthur Clark

Σχήμα 1.3: Gerty από την ταινία Moon

1.1.3 Εφαρμογές της τεχνητής νοημοσύνης σήμερα

Τα τελευταία χρόνια βλέπουμε μία αρκετά μεγάλη εξέλιξη στις εφαρμογές της ΤΝ όπως η ρομποτική, η μηχανική όραση, η μηχανική μάθηση και ο σχεδιασμός συμπεριφορών. Επίσης μεγάλη εξέλιξη γνωρίζει και η περιοχή των προγραμμάτων πρακτόρων (agents).

Γενικά δημιουργούνται πολλά ευφυή συστήματα. Υπάρχουν συστήματα πραγματικού χρόνου όπως το σύστημα MARVEL που εκτελεί ενέργειες σε διαστημόπλοια, μετά από επεξεργασία των δεδομένων που δέχεται και ενεργοποιεί κάποιο συναγερμό σε κατάσταση ανάγκης. Άλλα συστήματα όπως το ALVIN, μπορούν να ελέγξουν με επιτυχία ένα όχημα σε πραγματικές συνθήκες, με τη βοήθεια βιντεοκάμερας και άλλους αισθητήρες. Υπάρχουν επίσης συστήματα που μπορούν να διαγνώσουν ασθένειες και άλλα που μπορούν και ελέγχουν την εναέρια κυκλοφορία.

Σχήμα 1.4: Το σκυλάκι της SONY AIBO

Φυσικά πολλές εταιρίες αναπτύσσουν τα δικά τους συστήματα. Η SONY έχει αναπτύξει πολλά οικιακά ρομπότ όπως το σκυλάκι AIBO, το οποίο μπορεί να εκφράσει τα συναισθήματα του με την κίνηση και τον ήχο, αναγνωρίζοντας και την ομιλία. Η NASA μαζί με την Υπηρεσία Ανάπτυξης Προηγμένης Στρατιωτικής Τεχνολογίας των ΗΠΑ (DARPA), ανέπτυξαν ένα ρομπότ με το όνομα ROBONAUT. Ο «ρομποναύτης» είναι ειδικά φτιαγμένος για την συντήρηση του τηλεσκοπίου HUBBLE.

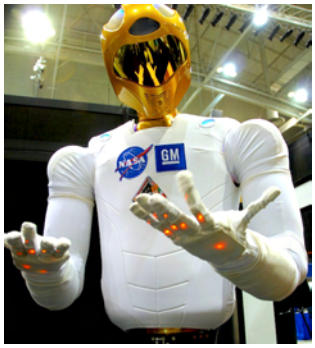
Επίσης υπάρχουν διάφορες εκδηλώσεις όπως είναι το ROBOCUP, που λαμβάνει χώρα κάθε χρόνο και περιλαμβάνει ρομπότ που προσπαθούν να παίξουν αυτόνομα ποδόσφαιρο. Σύμφωνα με το ερευνητικό πρόγραμμα MYLIFEBITS,

που αναπτύσσεται στη Microsoft, προσπαθούν να φτιάξουν έναν εικονικό εγκέφαλο. Μια βάση δηλαδή που θα κρατάει όλες τις εμπειρίες και γνώσεις ενός ατόμου.

1.2 Ρομποτική

1.2.1 Τι είναι η ρομποτική

Η ρομποτική είναι μια επιστήμη που ασχολείται με το σχεδιασμό, τη λειτουργία και τον έλεγχο ρομποτικών συστημάτων. Ένα ρομποτικό σύστημα αποτελείται από πολλά διαφορετικά τμήματα. Τα τμήματα αυτά συνήθως είναι το υλικό, δηλαδή η φυσική ύπαρξη μιας κατασκευής, το σύστημα έλεγχου και το κύριο λογισμικό. Έτσι, στο σχεδιασμό και στην υλοποίηση ενός ρομποτικού συστήματος συμμετέχουν επιστήμες όπως μηχανική, ηλεκτρονική και επιστήμη υπολογιστών, από την οποία χρησιμοποιείται κυρίος η τεχνητή νοημοσύνη και η θεωρία ελέγχου. Επίσης η βιολογία μπορεί να βοηθήσει με την έννοια ότι μπορεί να μας προσφέρει πληροφορίες για το πως επεξεργάζεται η φύση την πληροφορία.



Σχήμα 1.5: Ο RoboNaught της NASA.

1.2.2 Ιστορία της ρομποτικής

Η ιδέα για μηχανικά κατασκευάσματα που θα συμπεριφέρονται όπως ένας άνθρωπος δεν είναι καινούρια. Σύμφωνα με τη διαδικτυακή εγκυκλοπαίδεια Wikipedia η αναφορά σε αυτόματα ξεκινάει από αρχαίους χρόνους και βρίσκεται σε μυθολογίες αρχαίων πολιτισμών όπως την *Αρχαία Κίνα* [26] και την *Πτολεμαϊκή Αίγυπτο*. [11]

Η πρώτη εμφάνιση της λέξης ρομπότ προήλθε από τον Τσέχο συγγραφέα Karel Čapek στο έργο του R.U.R. (Rossum's Universal Robots). Έτσι από τη λέξη ρομπότ βγήκε και η ρομποτική. Σύμφωνα με το αγγλικό λεξικό της Οξφόρδης ο όρος ρομποτική αναφέρθηκε πρώτα από τον Ισαάκ Ασίμοφ, συγγραφέα επιστημονικής φαντασίας, σε μια ιστορία μικρού μήκους με τίτλο ''Liar!'', που εκδόθηκε το 1941. Ο Ασίμοφ δεν καταλάβαινε ότι χρησιμοποιούσε πρώτος τον όρο, πίστευε ότι έτσι όπως ονομάζουμε ηλεκτρονική την επιστήμη που έχει θέμα τον

ηλεκτρισμό, έτσι ονομάζουμε ρομποτική και την επιστήμη που ασχολείται με τα ρομπότ[13][12].

Ημερομηνία	**Σημασία**	**Όνομα Ρομπότ**	**Εφευρέτης**
3ος αι. π. Χ. και νωρίτερα	Μία από τις πρώτες περιγραφές αυτόματων εμφανίζεται στο κείμενο του *Lie Zi*. Γράφει για ένα συμβάν ανάμεσα στο βασιλιά Mu of Zhou (1023-957 π.Χ.) και το μηχανικό Yan Shi γνωστό ως «τεχνίτης» (artificer), ο οποίος παρουσίασε στον βασιλιά μια ανθρωπόμορφη μηχανική φιγούρα[26].		Yan Shi
420 π. Χ.	Ξύλινο πτηνό με έλικες που έπαιρνε ενέργεια μέσω ατμού και μπορούσε να πετάξει.		Αρχύτας ο Ταραντίνος
1ος αι. μ. Χ. και νωρίτερα	Περιγραφές περισσότερων από 100 αυτόματων και μηχανών συμπεριλαμβανομένων και μιας μηχανής φωτιάς, ενός οργάνου αέρα, μιας μηχανής που λειτουργεί με νομίσματα και μιας ατμομηχανής, στο *Pneumatica* και *Automata* από τον Heron της Αλεξάνδρειας.		Ctesibius, Philo του Βυζαντίου, Heron της Αλεξάνδρειας, και άλλοι

1206	Προγραμματιζόμενη μηχανική μπάντα και άλλα ανθρωπόμορφα αυτόματα.	Ρομποτική μπάντα, αυτόματο καθαρισμού χεριών [28], αυτοκινούμενα παγόνια [20]	Al-Jazari
1495	Σχέδια για ανθρωπόμορφα ρομπότ.	Μηχανικός Ιππότης	Leonardo da Vinci
1738	Μηχανική πάπια που μπορεί να φάει, να χτυπάει τα φτερά της και να κράζει.	Πάπια που Χωνεύει	Jacques de Vaucanson
1898	Ο Νικολά Τέσλα παρουσιάζει ένα τηλεκατευθυνόμενο σκάφος.	Τηλεαυτοματο	Νικολά Τέσλα
1921	Τα πρώτα φανταστικά αυτόματα με το όνομα «ρομπότ» εμφανίζονται στο έργο R.U.R.	Rossum's Universal Robots	Karel Čapek
Δεκαετία του '30	Ανθρωποειδές ρομπότ στην έκθεση World's Fairs το 1939 και 1940.	Elektro	Westinghouse Electric Corporation
1948	Απλά ρομπότ εκφράζουν βιολογικές συμπεριφορές[9].	Elsie και Elmer	William Grey Walter
1956	Πρώτο διαφημιστικό ρομπότ, για την εταιρία που ιδρύθηκε από τους George Devol και Joseph Engelberger, βασισμένο σε πατέντες του Devol[10].	Unimate	George Devol
1960	Το πρώτο ρομπότ με μηχανική δυαδική μνήμη.	Rudy	Michael Freeman Ph.D.
1961	Το πρώτο εγκατεστημένο βιομηχανικό ρομπότ.	Unimate	George Devol

1973	Το πρώτο βιομηχανικό ρομπότ με έξι ηλεκτρομηχανικώς οδηγούμενα τσεκούρια[31].	Famulus	KUKA Robot Group
1974	Ο πρώτος παγκόσμιος μικροϋπολογιστής ελεγχόμενος από ηλεκτρικό βιομηχανικό ρομπότ, που δόθηκε από την ASEA σε μια μικρή εταιρία στη Σουηδία. Το όνομα του ήταν IRB 6 και είχε πατενταριστεί από το 1972.	IRB 6	ABB Robot Group
1975	Το πρώτο ρομπότ που δημιούργησε προφορική έξοδο χρησιμοποιώντας λέξεις, προτάσεις και κομμάτια ομιλίας.	Leachim	Michael Freeman Ph.D.
1975	Προγραμματιζόμενο χέρι γενικής χρήσης.	PUMA	Victor Scheinman

Πίνακας 1.2: Μετάφραση του πίνακα για την ιστορία της ρομποτικής από την Wikipedia [3]

1.2.3 Εφαρμογές

Η ρομποτική έχει πληθώρα εφαρμογών. Μπορεί να είναι βιομηχανικές, ιατρικές, αλλά και για διαφημιστικούς σκοπούς. Στα βιομηχανικά ρομπότ μπορούμε να συναντήσουμε πολλά είδη βραχιόνων και άλλων ρομπότ παραγωγής. Τα πιο διαδεδομένα ρομποτικά χέρια είναι το Shadow Hand και το χέρι του RoboNaut[4]. Άλλα ρομπότ μπορεί να είναι ρομπότ κίνησης, στα οποία έχουμε κατηγορίες όπως ρομπότ με δυο ρόδες, ρομπότ με μια ρόδα ή ακόμα ρομπότ που στηρίζονται πάνω σε μια σφαίρα. Ένα πολύ γνωστό ρομπότ σε δύο ρόδες είναι το λεγόμενο ''Segway'' το οποίο φένεται στο σχήμα 1.7.

Στην πραγματικότητα τα είδη είναι αμέτρητα για να τα απαριθμήσουμε. Υπάρχουν αυτόνομα ρομπότ που μπορούν να πετάξουν, άλλα που είναι ικανά να περπατίσουν σαν τον άνθρωπο, μέχρι και ρομπότ που μπορούν να κάνουν ιστιοπλοΐα ή άλλα που μπορούν να παίξουν επιτραπέζια αντισφαίριση.

Σχήμα 1.6: Ένα ρομποτικό χέρι [8]

1.2.4 Επίπεδα αυτονομίας

Σύμφωνα με την Wikipedia υπάρχουν διάφορα επίπεδα αυτονομίας:

1. Απευθείας έλεγχος χρησιμοποιείται για ρομπότ που τηλεκατευθύνονται, στα οποία ο χρήστης έχει πλήρη έλεγχο.

2. Υποβοηθούμενη λειτουργία όπου ο χρήστης επιλέγει τους στόχους μεσαίου και υψηλού επιπέδου, τους οποίους το ρομπότ πρέπει να φέρει εις πέρας.

3. Ένα αυτόνομο ρομπότ μπορεί να μη χρειάζεται την επέμβαση του ανθρώπου για αρκετό καιρό. Για να υλοποιηθεί ένα αυτόνομο ρομπότ δεν είναι απαραίτητο να υλοποιηθούν και περίπλοκες εργασίες. Για παράδειγμα, τα ρομπότ σε γραμμές παραγωγής είναι αυτόνομα αλλά έχουν μια πολύ συγκεκριμένη λειτουργία.

Υπάρχει και ακόμα μια ταξινόμηση:

1. **Τηλεκατεύθυνση**. Ο άνθρωπος ελέγχει κάθε κίνηση. Κάθε αλλαγή στους ενεργοποιητές της μηχανής καθορίζεται πλήρως από τον χειριστή.

2. **Με επίβλεψη**. Ο άνθρωπος ελέγχει μόνο γενικές κινήσεις και αλλαγές στη θέση. Η μηχανή επιλέγει τις κινήσεις των ενεργοποιητών της.

3. **Αυτονομία σε επίπεδο στόχων**. Ο χειριστής καθορίζει τους στόχους και το ρομπότ προσπαθεί να τους πετύχει.

4. **Πλήρης αυτονομία**. Η μηχανή θα επιλέξει και θα πετύχει τους στόχους της χωρίς καμία παρέμβαση από άνθρωπο.

Σχήμα 1.7: Segway

1.3 Το θέμα της πτυχιακής εργασίας

Στην εν λόγω πτυχιακή εργασία σχεδιάσαμε και υλοποιήσαμε μία εφαρμογή για τον ευφυή έλεγχο ενός ρομπότ, την πλοήγηση του και τον εντοπισμό της θέσης του στο χώρο. Το βασικό πρόβλημα ήταν ο συνδιασμός ενός συστήματος, που θα κάνει την δουλειά του εντοπισμού μέσο κάμερας, με ένα σύστημα που θα ελέγχει το ρομπότ. Στην παρούσα εργασία, θεωρούμε ότι είναι ήδη υλοποιημένο το κομμάτι *πελάτη-εξυπηρετητή* για την επικοινωνία καθώς και το σύστημα εντοπίζει το στόχο μέσω κάμερας. Για το λόγο αυτό εδώ θα παρουσιαστεί ο βασικός αλγόριθμος που επεξεργάζεται τα δεδομένα του συστήματος εντοπισμού και στέλνει τις κατάλληλες εντολές στο ρομπότ για να κάνει τις ανάλογες κινήσεις.

Για το σύστημα χρησιμοποιήθηκε το ρομπότ PeopleBot[7] της εταιρίας MobileRobots[6] του οποίου τα χαρακτηριστικά μπορούν να βρεθούν στο παράρτημα Γ'. Η βιβλιοθήκη που χρησιμοποιήθηκε για τον προγραμματισμό της εφαρμογής είναι η ARIA (Advanced Robot Interface for Applications) και είναι γραμμένη σε C++, έτσι και η εφαρμογή είναι γραμμένη σε C++.

Στο δεύτερο κεφάλαιο περιγράφεται το λογισμικό που χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής καθώς και μια εκτενής έκθεση της βιβλιοθήκης ARIA και άλλων βοηθημάτων. Στο τρίτο κεφάλαιο παρουσιάζεται το σύστημα από θεωρητική άποψη αλλά και επεξηγείται πως υλοποιήθηκαν μερικά κομ-

μάτια του κώδικα. Στο τέταρτο κεφάλαιο υπάρχουν αναφορές και περιγραφές των πιο βασικών αλγορίθμων που παίρνουν μέρος στην εφαρμογή.

Στο τελευταίο κεφάλαιο αναφέρουμε τα συμπεράσματα στα οποία καταλήξαμε με την εκπόνηση της πτυχιακής εργασίας. Και στα παραρτήματα παρουσιάζεται ο κώδικα χωρισμένος σε διάφορα αρχεία, η σύνδεση της βιβλιοθήκης ARIA με το ολοκληρωμένο περιβάλλον ανάπτυξης (integrated development environment, IDE) Eclipse και τα χαρακτηριστικά του υλικολογισμικού.

Κεφάλαιο 2

Περιγραφή λογισμικού

2.1 MobileRobots

Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε η πλατφόρμα ARIA (Advanced Robot Interface for Applications). Η πλατφόρμα ARIA έχει δημιουργηθεί από την ομάδα Adept MobileRobots. Ο στόχος της εταιρίας MobileRobots είναι να σχεδιάζει και να κατασκευάζει ρομποτικά συστήματα για ερευνητικό έργο, αλλά και για άλλους σκοπούς, όπως εμπορική χρήση ή για προγραμματιστές που φτιάχνουν διαφημιστικές εφαρμογές. Το πρώτο ρομπότ ανακοινώθηκε το 1995, και από τότε μέχρι σήμερα έχει πραγματοποιηθεί μεγάλη πρόοδος. Πλέον, είναι από τις μεγαλύτερες εταιρίες σχεδιασμού και παραγωγής έξυπνων ρομποτικών συστημάτων. Έτσι, χρησιμοποιώντας το PeopleBot και την πλατφόρμα ARIA, καταφέραμε να φτιάξουμε ένα έξυπνο ρομποτικό σύστημα, το οποίο να μπορεί να έχει γνώση της θέσης του στον χώρο, ενώ ταυτόχρονα είναι ικανό να ακολουθεί ένα συγκεκριμένο στόχο.

2.2 ARIA

Στο σημείο αυτό θα αναλυθεί η πλατφόρμα ARIA. Η ARIA είναι μια βιβλιοθήκη γραμμένη στην γλώσσα προγραμματισμού C++, με την οποία μπορεί κάποιος να ελέγξει δυναμικά την ταχύτητα, την περιστροφή, την πορεία, τη σχετική πορεία και άλλες μεταβλητές κίνησης ενός ρομπότ. Αυτό το πετυχαίνει είτε μέσω των χαμηλού επιπέδου εντολών, είτε μέσω του υψηλού επιπέδου συστήματος γνωστού και ως ''Actions''. Επίσης λαμβάνει πληροφορίες από τα διάφορα αισθητήρια όργανα που πιθανόν να έχει ένα ρομπότ.

Εκτός από την βιβλιοθήκη ARIA, υπάρχει και η βιβλιοθήκη ArNetworking. Με τη βιβλιοθήκη ArNetworking μπορεί κάποιος να πραγματοποιήσει επικοινωνία ανάμεσα σε έναν *πελάτη* και στον *εξυπηρετητή* που τρέχει τοπικά στο ρομπότ. Με τον τρόπο αυτό, μπορεί να επιτευχθεί απομακρυσμένος έλεγχος καθώς και μετάδοση κάθε είδους πληροφορίας. Αρκεί να χρησιμοποιούν την βιβλιοθήκη ArNetworking και ο *εξυπηρετητής* και ο *πελάτης*. Συνεπώς, γίνεται φανερό ότι μπορεί να υπάρξει διεπαφή *πελάτη - εξυπηρετητή*. Ένας εύκολος τρόπος να χρησιμοποιήσεις αυτή τη δυνατότητα είναι μέσω του έτοιμου πελάτη

της MobileRobots, MobileEyes, που έχει γραφικό περιβάλλον και είναι εύκολος ως προς τη χρήση.

Σχήμα 2.1: MobileRobots' PeopleBot

Πολλά εργαλεία συμπεριλαμβάνονται στην ARIA καθώς επίσης και σε άλλες παρεμφερείς βιβλιοθήκες. Με τα εργαλεία αυτά μπορεί να επιτευχθεί σύνθεση και αναγνώριση ομιλίας, αναπαραγωγή ήχου, μαθηματικές συναρτήσεις και άλλα. Επίσης, η βιβλιοθήκη είναι γραμμένη σε C++ αλλά είναι εφικτή η ενεργοποίηση εφαρμογών σε Java και σε Python, καθώς και η εγκατάσταση και επιτυχής εφαρμογή μέσω των λεγόμενων ''wrappers''. Η ARIA υποστηρίζεται από τα λειτουργικά συστήματα GNU/Linux[2] και Windows[1]. Επιπροσθέτως, η βιβλιοθήκη σαν πηγαίος κώδικας υπόκειται στην άδεια GNU General Public License. Αυτό σημαίνει ότι διανέμεται ελεύθερα, ενώ οποιαδήποτε αλλαγή και μετέπειτα διανομή της, θα πρέπει να υπακούει σε αυτή την άδεια και να είναι ελεύθερη. Υπάρχει αρκετά εκτεταμένη τεκμηρίωση και πολλά παραδείγματα που μπορούν να βοηθήσουν τον προγραμματιστή.

Μερικές προγραμματιστικές ιδιαιτερότητες που εμφανίζονται στην βιβλιοθήκη ARIA:

- Τα ονόματα των κλάσεων ξεκινάνε πάντα με ''Ar'' και έχουν ανάμεικτα κεφαλαία και μικρά.
- Τα ονόματα από τις σταθερές ξεκινάνε με κεφαλαίο ή είναι γραμμένες καθαρά με κεφαλαία.
- Τα ονόματα των μεταβλητών που ανήκουν σε μια κλάση ξεκινάνε με το πρόθεμα ''my''.
- Τα ονόματα των στατικών μεταβλητών μιας κλάσης έχουν πρόθεμα ''our''.

- Οι μέθοδοι που ανήκουν σε μια κλάση ξεκινάνε με μικρό γράμμα.
- Με κεφαλαίο γράμμα ξεκινάει κάθε λέξη μετά την πρώτη σε μια μέθοδο ή μεταβλητή. π. χ. thisIsAnExample
- Κάθε κλάση μπορεί να χρησιμοποιηθεί σε πρόγραμμα με πολλαπλά νήματα, αρκεί να υπάρχει κάποια μέθοδος ασφαλείας. Για παράδειγμα τα ''mutexes''.

2.2.1 Σχέση πελάτη εξυπηρετητή ARIA-Robot

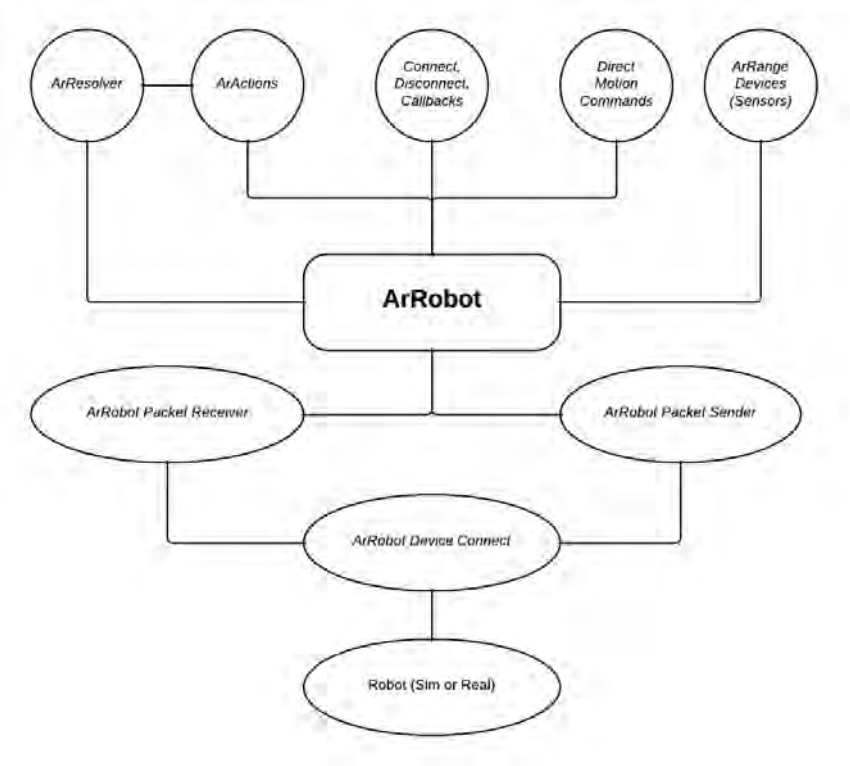
Οι διεργασίες του κεντρικού *εξυπηρετητή* έχουν υλοποιηθεί στο υλικολογισμικό του λειτουργικού συστήματος του AmigoBot (ARCOS, AROS, AmigOS, και άλλα), οι οποίες τρέχουν στο μικροελεγκτή του ρομπότ. Αυτές οι διεργασίες έχουν να κάνουν με τον έλεγχο και τη λειτουργία εργασιών χαμηλού επιπέδου. Εξαιτίας αυτού, οι εργασίες αυτές πρέπει να γίνουν σε πολύ συγκεκριμένα χρονικά διαστήματα και είναι πολύ κρίσιμες. Τέτοιες λειτουργίες είναι συνήθως ο έλεγχος κινήσεων και πορείας καθώς και η εκτίμηση της θέσης του ρομπότ μέσο οδομετρίας. Ακόμη, η ανάγνωση πληροφορίας γίνεται από τα αισθητήρια όργανα. Όλα αυτά τα κομμάτια, δηλαδή το ρομπότ, το υλικολογισμικό, ο μικροελεγκτής και οι αισθητήρες συνήθως ονομάζονται «ρομποτική πλατφόρμα».

Βέβαια το υλικολογισμικό δεν μπορεί να επεξεργαστεί λειτουργίες υψηλού επιπέδου. Αυτή τη δουλειά, τις υψηλού επιπέδου λειτουργίες δηλαδή, τις εκτελεί ένας *πελάτης* που είναι εγκατεστημένος σε έναν προσωπικό υπολογιστή. Οι συγκεκριμένες λειτουργίες συμπεριλαμβάνουν αναγνώριση και αποφυγή εμποδίων, συνδυασμό της πληροφορίας από τους αισθητήρες, έξυπνη πλοήγηση, σύνθετη κίνηση PTZ (Pan, Tilt, Zoom) κάμερας, εντοπισμό θέσης, χαρτογράφηση και πολλά άλλα. Στόχος λοιπόν της διεπαφής ARIA είναι να κάνει εφικτή την επικοινωνία ανάμεσα στην «ρομποτική πλατφόρμα» και τον απομακρυσμένο λογισμικό *πελάτη*.

Το κεντρικό κομμάτι της ARIA είναι η κλάση ArRobot . Είναι η κλάση που ελέγχει τον κύκλο επικοινωνίας με το υλικολογισμικό, δέχεται και στέλνει πληροφορίες σχετικές με την κατάσταση του ρομπότ, ενεργοποιεί λειτουργίες και στέλνει πίσω σχετικές εντολές. Παράλληλα χρησιμοποιείται και με αντικείμενα άλλων κλάσεων και γενικές συναρτήσεις που έχουν να κάνουν με το ρομπότ. Η υποδομή με τα λεγόμενα ''Actions'' βοηθάει στη συνδυασμένη συμπεριφορά του συστήματος, δηλαδή μπορούν να συνδυαστούν λειτουργίες που είναι ανεξάρτητες μεταξύ τους, σε μια γενική λειτουργία. Τέτοιες λειτουργίες είναι η αυτόματη πλοήγηση, η τηλεκατεύθυνση και άλλες.

2.2.2 Επικοινωνία ρομπότ

Η πιο σημαντική διαδικασία, που πρέπει να γίνει και πρώτη, είναι να συνδεθεί η εφαρμογή *πελάτης* με το υλικολογισμικό του ρομπότ. Ύστερα μπορούμε να συνδέσουμε κάθε είδους συσκευή για να βρίσκουμε την απόσταση, κάποια κάμερα, το βραχίονα ίσως ή γενικά οποιοδήποτε άλλο αντικείμενο χρησιμοποιεί



Σχήμα 2.2: Η αρχιτεκτονική της βιβλιοθήκης Aria

σαν αναφορά το αντικείμενο ArRobot. Παρακάτω παρουσιάζονται μερικές ενδεικτικές διατάξεις σύνδεσης.

Όπως βλέπουμε υπάρχει η δυνατότητα να συνδέσουμε τον πελάτη με έναν προσομοιωτή. Με αυτόν τον τρόπο έγινε το μεγαλύτερο μέρος της αποσφαλμάτωσης πριν μεταφερθούμε σε πλήρη κλίμακα. Βεβαίως η προσομοίωση με τον πραγματικό κόσμο διαφέρει κατά πολύ και όταν πλέον κινηθήκαμεε σε πλήρη κλίμακα, έπρεπε και εκεί να γίνουν πολλές διορθώσεις. Το αντικείμενο που πραγματοποιεί τη σύνδεση με τον μικροελεγκτή του ρομπότ ανήκει στην κλάση **ArRobotConnector** και έχει κάποια ορίσματα τα οποία διαβάζει από ένα αρχείο, ή μπορεί να του δοθούν κατευθείαν από τη γραμμή εντολών. Στόχος του ArRobotConnector είναι να ψάξει στην αρχή κάποια τοπική σύνδεση, μήπως υπάρχει κάποιος ενεργός προσομοιωτής. Αν δεν πετύχει αυτό προσπαθεί να συνδεθεί σε μία από τις σειριακές θύρες. Για την πραγματοποίηση μιας απομακρυσμένης σύνδεσης, θα πρέπει να παραχωρηθούν οι ανάλογες παράμετροι στον **ArRobotConnector** είτε από το αρχείο, είτε από την γραμμή εντολών. Για να λειτουργήσει επιτυχώς όλη η παραπάνω διαδικασία, πρέπει να τεθεί η μέθοδος **ArArgumentParser::loadDefaultArguments()**, για να φορτωθούν οι προεπιλεγμένες παράμετροι από το αρχείο Aria.args ή από τη μεταβλητή συστήματος ARIAARGS.

2.2.3 Η κλάση ArRobot

Όπως αναφέρθηκε και παραπάνω η διεπαφή ARIA λειτουργεί με το μοντέλο *πελάτη εξυπηρετητή*. Εν προκειμένω, ο *πελάτης* είναι είτε ο ενσωματωμένος στο ρομπότ ηλεκτρονικός υπολογιστής, είτε κάποιος απομακρυσμένος Η/Υ και ο *εξυπηρετητής* είναι το υλικολογισμικό του ρομπότ. Η επικοινωνία γίνεται μέσω *πακέτων*. Τα *Πακέτα Πληροφορίας Εξυπηρετητή*(Server Information Packets, SIPs), είναι τα βασικά πακέτα που στέλνονται κάθε εκατό χιλιοστά του δευτερολέπτου από τον *εξυπηρετητή*. Αυτά τα πακέτα περιέχουν πληροφορίες από τους αισθητήρες όσον αφορά την τάση της μπαταρίας, την ταχύτητα, τη θέση και άλλα. Όλα αυτά τα δεδομένα αποθηκεύονται στο *Είδωλο Κατάστασης* και μπορούμε να τα προσπελάσουμε με μεθόδους του αντικειμένου ArRobot.

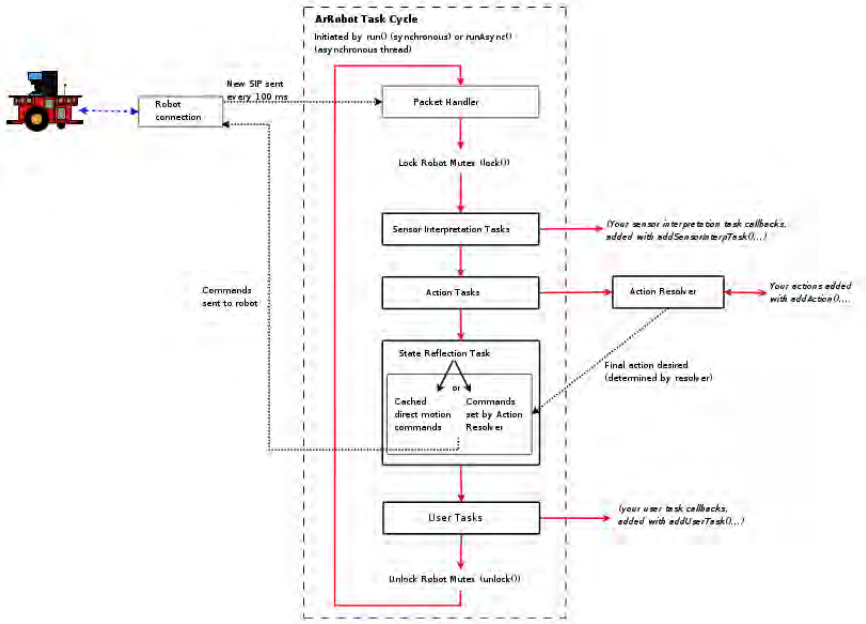
Υπάρχουν και τα *εκτεταμένα πακέτα* τα οποία έχουν διαφορετικό τύπο δεδομένων. Μπορούν να δημιουργηθούν και δικά μας πακέτα και δικοί μας *χειριστές πακέτων* προσθέτοντάς τους με την μέθοδο **ArRobot::addPacketHandler()**. Τέλος, έχουμε τα *πακέτα εντολών* τα οποία χρησιμοποιούνται για να κινηθούν το ρομπότ. Τέτοια πακέτα στέλνονται στον *εξυπηρετητή* όταν χρησιμοποιούμε στον κώδικα μας *Συναρτήσεις Εντολών Κίνησης (Motion Command Functions)*, τα υψηλού επιπέδου ''*Actions*'' ή σε πολύ βασικό επίπεδο τις *Άμεσες Εντολές*(Direct Commands). Απαιτείται μεγάλη προσοχή στη χρήση τέτοιων λειτουργιών, γιατί τα πακέτα εντολών μπορεί να μπερδευτούν και να εμποδίζονται αναμεταξύ τους. Το εν λόγω θέμα θα αναλυθεί εκτενέστερα στην πορεία.

Ο κύκλος συγχρονισμού του ρομπότ ενεργοποιείται κάθε φορά που έρχεται ένα καινούριο πακέτο SIP αν δεν υπάρξει κάποιο πρόβλημα από την πλευρά του υλικολογισμικού. Έτσι ο κύκλος περνάει από όλα τα στάδια ανα τακτά χρονικά διαστήματα και κρατάει τη ρομποτική πλατφόρμα ενήμερη. Για την εκκίνιση αυτού του κύκλοου πρέπει να κληθεί η μέθοδος **ArRobot::run()** για να τρέχει ο κύκλος σε συγχρονισμό με το υλικολογισμικό. Εναλλακτικά, είναι δυνατόν να κληθεί η μέθοδος **ArRobot::runAsync()** για ασύγχρονη λειτουργία. Στην εκκίνηση του κύκλου το **ArRobot** κλειδώνει το σημαφόρο του για να προστατέψει τα δεδομένα του από διαφορετικά νήματα εκτέλεσης. Μετά την κλήση της μεθόδου, που ξεκινάει τον κύκλο συγχρονισμού του ρομπότ, είναι αναγκαίο να κλειδώνεται ο σημαφόρος κάθε φορά που γίνεται επεξεργασία δεδομένων της κλάσης **ArRobot**.

Το Είδωλο Κατάστασης του ρομπότ αντιπροσωπεύει ένα στιγμιότυπο της λειτουργικότητας του έτσι ώστε να γίνεται σωστός έλεγχος. Τα δεδομένα μπορούν να προσπελαστούν με τις μεθόδους της κλάσης **ArRobot**:

ArRobot::getPose(), ArRobot::getX(), ArRobot::getY(), ArRobot::getTh(), ArRobot::getVel(), ArRobot::getRotVel(), ArRobot::getBatteryVoltage(), ArRobot::isLeftMotorStalled(), ArRobot::isRightMotorStalled(), ArRobot::getCompass(), ArRobot::getAnalogPortSelected(), ArRobot::getAnalog(), ArRobot::getDigIn(), ArRobot::getDigOut().

Το πακέτο SIP παρέχει και πληροφορίες για τις μετρήσεις του αισθητήρα υπερήχων, οι οποίες μπορούν να προσπελαστούν με τις εξής μεθόδους:



Σχήμα 2.3: Μία γενική εικόνα του ρομποτικού κύκλου.

ArRobot::getNumSonar(), ArRobot::getSonarRange(), ArRobot::isSonarNew(), ArRobot::getSonarReading(), ArRobot::getClosestSonarRange(), ArRobot::getClosestSonarNumber(). Όπως επίσης και από τις μεθόδους που ανήκουν στην κλάση**ArSonarDevice**. Το είδωλο κατάστασης επίσης χρησιμοποιείται για να στείλει προηγούμενες εντολές κίνησης, έτσι ώστε να αντιστοιχούν στα ανανεωμένα δεδομένα κατάστασης.

2.2.4 Ελέγχοντας το ρομπότ με Εντολές και Δράσεις

Ο ARIA εξυπηρετητής μπορεί να ελέγξει την κίνηση του ρομπότ με *Άμεσες Εντολές, Εντολές Κίνησης ή Δράσεις*. Καθίσταται αναγκαία η παράθεση όλων των δυνατοτήτων ελέγχου κίνησης από το χαμηλότερο επίπεδο προς το υψηλότερο. Στο χαμηλότερο επίπεδο υπάρχουν οι άμεσες εντολές, οι οποίες συνήθως αποτελούνται από έναν αριθμό του ενός byte και ίσως από κάποια ορίσματα. Για να σταλούν τέτοιες εντολές είναι δυνατόν να κληθεί η μέθοδος **ArRobot::com()** για εντολές χωρίς ορίσματα. Διαφορετικά μπορεί να κληθεί η μέθοδος **ArRobot::comInt()** για εντολές που έχουν ένα όρισμα, όπως η εντολή ENABLE. Με την τελευταία εντολή, εάν σταλούν μαζί 1 ή 0 σαν όρισμα, τότε το υλικολογισμικό του ρομπότ θα ενεργοποιήσει ή θα απενεργοποιήσει τα μοτεράκια των τροχών αντίστοιχα. Κάθε εντολή έχει μια αντίστοιχη σταθερά που μπορεί να βρεθεί στον καταμετρητή **ArCommands**, όπως παραδείγματος χάριν **ArCommands::ENABLE**. Παρόλα αυτά, απαιτείται ιδιαίτερη προσοχή με τις άμεσες εντολές γιατί διαφέρουν από ρομπότ σε ρομπότ.

Μετά τις άμεσες εντολές, στο αμέσως επόμενο επίπεδο, ακολουθούν οι εντολές κίνησης. Οι εντολές κίνησης ελέγχουν κατευθείαν τιμές από το είδωλο κατάστασης του ρομπότ και στέλνονται οι ανάλογες εντολές όταν ελεγχθεί στον κύκλο συγχρονισμού το είδωλο κατάστασης του ρομπότ. Τέτοιες εντολές μπορεί να είναι **ArRobot::setVel(), ArRobot::setRotVel(), ArRobot::setHeading, ArRobot::move(), ArRobot::stop()**. Γίνεται λοιπόν, φανερό, ότι μπορούν να ελεχθούν η ταχύτητα, η κυκλική ταχύτητα, η σχετική γωνία πορείας ενώ ταυτόχρονα είναι δυνατή η ακύρωση όλων των άλλων για να σταματήσει το ρομπότ να κινείται. Οι εντολές κίνησης, από την άλλη, χρειάζονται προσοχή, γιατί μπορεί να γίνει σύγχυση εντολών υψηλότερου επιπέδου. Δηλαδή θα πρέπει να καλείται η μέθοδος **ArRobot::clearDirectMotion()**, ώστε να μπορεί κάποια δράση να πάρει τον έλεγχο.

Οι εντολές του υψηλότερου επιπέδου είναι οι *Δράσεις*. Για να ενσωματωσουθούν πιο σύνθετες και περίπλοκες κινήσεις θα πρέπει να χρησιμοποιηθεί το σύστημα των δράσεων. Οι δράσεις είναι αντικείμενα της κλάσης **ArAction**. Με πολύ εύκολο και άμεσα υλοποιήσιμο κώδικα μπορούν να γίνουν περίπλοκοι συνδυασμοί κινήσεων. Μπορούν να υλοποιηθούν ξεχωριστά οι επιθυμητές κινήσεις σαν δράσεις και μετά να προστεθούν στο αντικείμενο του ρομπότ με τη μέθοδο **ArRobot::addAction()**, δίνοντας μαζί σαν όρισμα τον συντελεστή σημαντικότητας τις δράσεις. Έτσι ο χειριστής των δράσεων την κατάλληλη στιγμή, θα περάσει έναν συνδυασμό εντολών κίνησης στο είδωλο κατάστασης του ρομπότ και από εκεί θα εκτελεστούν από το υλικολογισμικό του ρομπότ. Εμείς δοκιμάσαμε τις δράσεις αλλά δεν τις χρησιμοποιήσαμε γιατί δεν ήταν τόσο περίπλοκο να κάνουμε άμεσες κινήσεις, γιατί θέλαμε καλύτερο έλεγχο των κινήσεων και γιατί υπήρχαν κάποια προβλήματα με τον συγχρονισμό. Για το λόγο αυτο, δε θα αναφερθούν περισσότερα για τις δράσεις. Περισσότερα μπορούν να βρεθούν στην τεκμηρίωση της διεπαφής ARIA[5].

2.3 Συσκευές Εμβέλειας

Σύμφωνα με την τεκμηρίωση της ARIA οι συσκευές εμβέλειας **ArRangeDevice** είναι αφαιρετικές μορφές των αισθητήρων του ρομπότ, παίρνουν μετρήσεις για το περιβάλλον ανα τακτά χρονικά διαστήματα και έχουν τη δυνατότητα να εντοπίζουν εμπόδια και διάφορα άλλα υλικά που βρίσκονται στο χώρο. Οι μετρήσεις μεταφράζονται σε σημεία στον γενικό σύστημα συντεταγμένων. Το σύστημα συντεταγμένων αποτελείται από δυο άξονες.

Οι κύριες υλοποιήσεις συσκευών εμβέλειας(**ArRangeDevice**) που υποστηρίζει η ARIA είναι:

- Το σόναρ (**ArSonarDevice**)
- Το λέιζερ (**ArLazer**)
- Οι προφυλακτήρες (**ArBumpers**)
- Και οι υπέρυθρες (**ArIRs** που είναι με τέτοιο τρόπο τοποθετημένες έτσι

ώστε να εντοπίζονται εμπόδια που δεν μπορούν να τα εντοπίσουν τα σόναρ).

Άλλες συσκευές τρισδιάστατης εμβέλειας και η κάμερα, υποστηρίζονται από διαφορετικό λογισμικό. Επίσης υπάρχει μια εικονική συσκευή εμβέλειας απαγορευμένων περιοχών(**ArForbiddenRangeDevice**) που προορίζεται για να εντοπίζονται εικονικές απαγορευμένες περιοχές στον χάρτη(**ArMap**).

Οι συσκευές εμβέλειας συνδέονται με ένα αντικείμενο **ArRobot** έτσι ώστε να επικοινωνεί με κάποια συγκεκριμένη συσκευή του ρομπότ. Μπορεί να προστεθεί μια συσκευή εμβέλειας στην εφαρμογή καλώντας την μέθοδο **ArRobot::addRangeDevice()** και να αφαιρεθεί καλώντας τη μέθοδο **ArRobot::remRangeDevice()**. Μπορούν να γίνουν επίσης ερωτήματα στη λίστα με τις συσκευές καλώντας τις μεθόδους **ArRobot::findRangeDevice(), ArRobot::hasRangeDevice**, ή για να τεθούν ερωτήματα σε όλη τη λίστα καλείται η μέθοδος **ArRobot::getRangeDeviceList()**. Κάθε συσκευή εμβέλειας έχει το δικό της σημαφόρο, για να μπορούν να επεξεργαστούν τα δεδομένα από διαφορετικά νήματα εκτέλεσης.

2.4 Χάρτης

Συνήθως σε μια ρομποτική εφαρμογή θα χρειαστεί να αποθηκεύονται διάφορα δεδομένα για το περιβάλλον. Η ARIA μας παρέχει με μια κλάση ArMap για να διαβάζονται δεδομένα από ένα αρχείο χάρτη. Ένας χάρτης συνήθως περιέχει δεδομένα όπως τοίχοι, εμπόδια και άλλα δυναμικά δεδομένα, όπως κάποιοι στόχοι στο χώρο, απαγορευμένες περιοχές και άλλα. Ο χάρτης είναι σημαντικό στοιχείο για την εφαρμογή, διότι χωρίς αυτόν η αναγνώριση θέσης και η αυτόματη πλοήγηση θα ήταν αδύνατες.

2.5 ARIA Navigation and Localization (ARNL)

Η βιβλιοθήκη ARNL[6] είναι μια προσθήκη στην βιβλιοθήκη ARIA έτσι ώστε να δώσει τ η δυνατότητα στο ρομπότ να μπορεί να κινείται έξυπνα στον χώρο και να εντοπίζει τη θέση του. Συνδυάζει τις πληροφορίες μαζί με τις οδομετρικές πληροφορίες και τον χάρτη που έχει κατασκευαστεί για να εντοπιστεί η θέση του ρομπότ. Το κομμάτι που υλοποιεί την πλοήγηση χρισημοποιεί τη θέση του ρομπότ και τον χάρτη για να βρει ένα ασφαλές μονοπάτι, έτσι ώστε το ρομπότ να φτάσει στον στόχο του.

Υπάρχουν τέσσερα διαφορετικά πακέτα/βιβλιοθήκες που έχουν αντιστοιχία με τους αισθητήρες του ρομπότ για το οποίο θα προγραμματιστεί μια εφαρμογή. Η βασική βιβλιοθήκη είναι η ''BaseArnl'' η οποία περιέχει τη λειτουργία για το σχεδιασμό διαδρομής και κάποια βασικά χαρακτηριστηκά που χρησιμοποιούνται από τις βιβλιοθήκες για τον εντοπισμό της θέσης. Η βιβλιοθήκη ''Arnl'' περιέχει την *κλάση* που χρησιμοποιεί ένα λέιζερ σαρώνοντας το χώρο. Υπάρχει η βιβλιοθήκη ''SonArnl'' που χρησιμοποιήθηκε και από εμάς γιατί εντοπίζει την

θέση του ρομπότ με πληροφορίες απο τα σόναρ του ρομπότ. Επίσης χρησιμοποιήθηκε το PeopleBot[7] που είχαμε στη διάθεση μας και είχε μόνο τα σόναρ. Τέλος έχουμε την βιβλιοθήκη ``Mogs που κάνει τον εντοπισμό μέσω συστήματος GPS.

Για να υλοποιήσουμε ένα πρόγραμμα που εκμεταλλεύεται τις παραπάνω δυνατότητες, πρέπει να δημιουργήσουμε ένα αντικείμενο της *κλάσης* **ArLocalizationTask**. Η *κλάση* αυτή υλοποιεί τον αλγοριθμικό εντοπισμό θέσης του Monte Carlo για να τοποθετήσει το ρομπότ στη σωστή θέση πάνω στον χάρτη που του δενεται. Για να γραφτεί ένα πρόγραμμα που να περιέχει αυτές τις δυνατότητες πρέπει να συμπεριληφθούν τα εξής αρχεία στον κώδικά μας:

Arnl/Aria.h, Arnl/Arnl.h, Arnl/ArLocalizationTask.h, Arnl/ArNetworking.h και να συνδεθεί ο πηγαίος κώδικας με τις βιβλιοθήκες **libBaseArnl, libAriaForArnl, libAriaNetworkingForArnl, Arnl/lib**.

Για να λειτουργήσουν όλα αυτά το πρώτο που δημιουργείται είναι το αντικείμενο της **ArLocalizationTask** και του περνώνται σαν ορίσματα το ρομπότ, το σόναρ και ο χάρτης.

```
1 // Create the localization task (it will start its own thread here)
2 ArLocalizationTask myLocTask(myRobot, myLaser, mapname);
```

Και μετά βρίσκεται η αρχική θέση του ρομπότ.

```
1 // Set the initial pose to the robot's "Home" position from the map,
   or
2 // (0,0,0) if none, then let the localization thread take over.
3 myLocTask.localizeRobotAtHomeBlocking();
```

Από εκεί και πέρα αναλαμβάνει το νήμα της **ArLocalizationTask** και ανανεώνει κάθε φορά που κινείται το ρομπότ. Αυτό το ρομπότ μπορεί να ελεγχθεί με κάποιες παραμέτρους. Οι προκαθορισμένες τιμές είναι $\pm 200_{mm}$ μετατόπιση και $\pm 5^{\circ}$ περιστροφή.

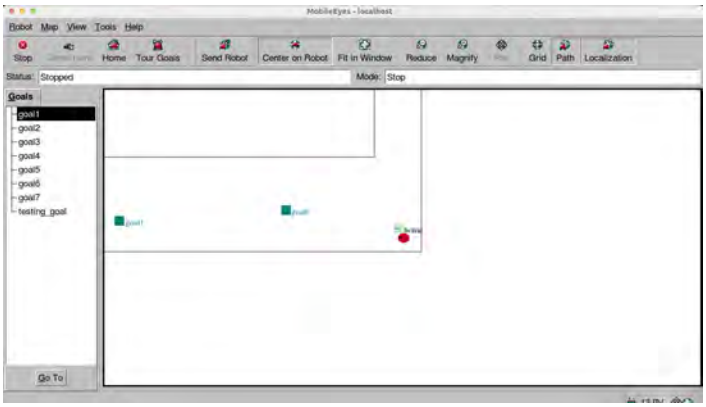
2.6 MobileEyes

Το MobileEyes είναι ένα γραφικό περιβάλλον που παρέχει πολλές δυνατότητες για τον έλεγχο του ρομπότ. Είναι ένας *πελάτης* που συνδέεται με τον *εξυπηρετητή* του ρομπότ και από εκεί μπορούμε να ελέγξουμε την πλοήγηση και άλλες λειτουργίες του ρομπότ. Το χρισημοποιήσαμε για λόγους αποσφαλμάτωσης γιατί μας δίχνει εύκολα και γρίγορα πληροφορίες για το ρομπότ. Για να λειτουργήσει πρέπει κάποιος να χρισημοποιήσει την βιβλιοθήκη **ArNetworking**.

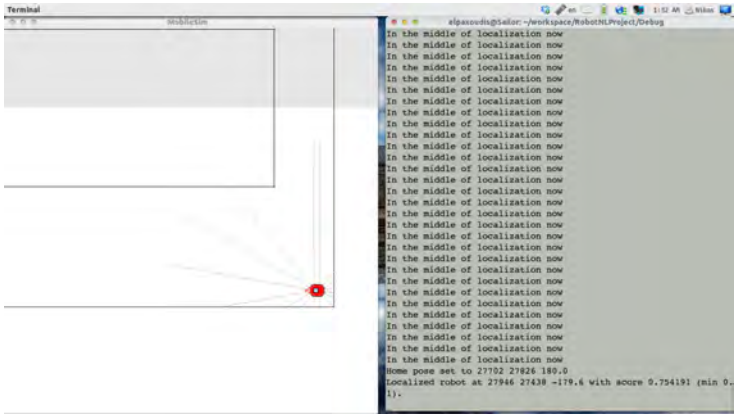
2.7 MobileSim

Το MobileSim είναι ένα περιβάλλον προσομοίωσης για αποσφαλμάτωση και πειραματισμό για προγράμματα που χρησιμοποιούν την βιβλιοθήκη ARIA. Το

MobileSim παρέχει τη δυνατότητα να δεχτεί συνδέσεις σε μια TCP πόρτα αντί για την σειριακή και έτσι είναι εύκολο να δοκιμάζει κάποιος τις εφαρμογές και να τις τρέχει μετά στο κανονικό ρομπότ, χωρίς να γίνει κάποια αλλαγή στον κώδικα.



Σχήμα 2.4: Διεπαφή MobileEyes.



Σχήμα 2.5: Το περιβάλλον προσομοίωσης.

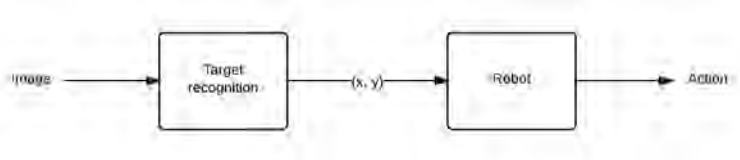
Κεφάλαιο 3

Περιγραφή του συστήματος

Σε αυτό το κεφάλαιο θα αναφερθούν όλα τα συστήματα που χρησιμοποιήθηκαν αναλυτικότερα, ενώ ταυτόχρονα θα εξεταστεί ο στόχος καθώς και τα σημεία της τελικής επιτυχίας.

3.1 Στόχος

Στόχος ήταν να συνδυαστεί ένα σύστημα αναγνώρισης αντικειμένων μέσω κάμερας, με ένα σύστημα πλοήγησης του ρομπότ, έτσι ώστε να δημιουργηθεί μια εφαρμογή με την οποία το ρομπότ θα ακολουθεί αυτό το συγκεκριμένο στόχο. Θεωρήθηκε ότι το σύστημα της αναγνώρισης μέσω κάμερας είναι ήδη υλοποιημένο, οπότε μένει μόνο να υλοποιηθεί η διαδικασία που θα καθοδηγεί το ρομπότ αποδοτικά και με ασφάλεια στο χώρο, έτσι ώστε να ακολουθεί τον στόχο του.



Σχήμα 3.1: Αφαιρετικό διάγραμμα συστήματος

3.2 Θεωρητικό Μοντέλο

Νοήθηκε ένα μοντέλο το οποίο θα αποτελείται από τρεις κύριες καταστάσεις. Ανάλογα με την εκάστοτε περίσταση, θα εναλλάσσεται ανάμεσα σε αυτές τις καταστάσεις για να ακολουθεί τον στόχο του. Η κατάληξη σε αυτές τις καταστάσεις προήλθε από την απλούστατη σκέψη του τι πιθανά σενάρια μπορούμε να συναντήσουμε. Έτσι λοιπόν τα πιο πιθανά σενάρια μπορεί να είναι:

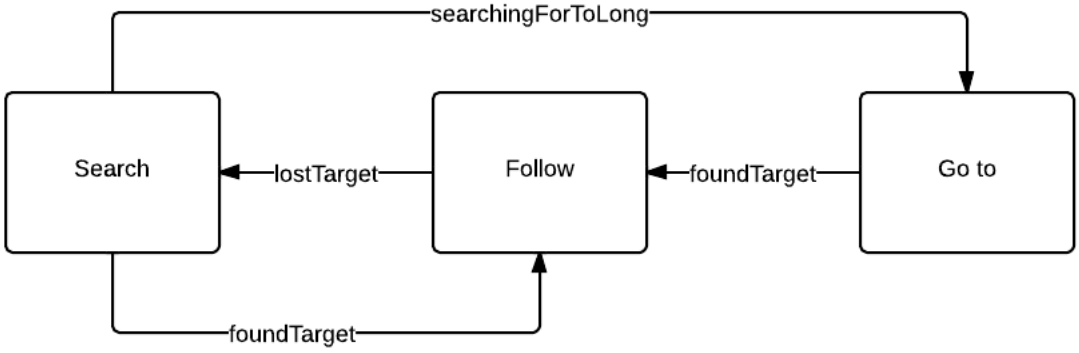
- Ο στόχος να είναι εντός οπτικής εμβέλειας.
- Ο στόχος να είναι εκτός οπτικής εμβέλειας.

- Ο στόχος να είναι εκτός οπτικής εμβέλειας, για περισσότερο χρόνο από τον αναμενόμενο.

Έτσι μπορούν να εξαχθούν οι τρεις καταστάσεις οι οποίες είναι:

- *Ακολουθώ (Following)*
- *Ψάχνω (Searching)*
- *Πηγαίνω (Going to)*

Πιο επίσημα, στο σχήμα 3.2 φαίνονται οι τρεις καταστάσεις και με ποιο τρόπο συνδέονται. Δηλαδή στην κατάσταση όπου το ρομπότ ακολουθεί τον στόχο, εάν γίνει κάτι με αποτέλεσμα την απώλεια του στόχου, τότε γίνεται μετάβαση στην κατάσταση της αναζήτησης. Στην κατάσταση της αναζήτησης, εάν ο στόχος είναι ξανά εντός εμβέλειας, τότε μεταβαίνουμε πάλι στην κατάσταση που το ρομπότ ακολουθεί το στόχο. Επίσης για να γίνει λίγο πιο έξυπνο, αν η αναζήτηση διαρκέσει περισσότερη από την επιθυμητή ώρα, τότε το ρομπότ αρχίζει να πηγαίνει σε συγκεκριμένα σημεία στον χάρτη με μια συγκεκριμένη σειρά. Και τέλος, στην κατάσταση που το ρομπότ πηγαίνει σε αυτά τα σημεία και κάπου στην πορεία του συναντήσει το στόχο, ξαναγυρνάει στην κατάσταση που ακολουθεί το στόχο τροφοδοσία.



Σχήμα 3.2: Οι τρείς κύριες καταστάσεις

3.3 Αντικειμενοστρεφής προγραμματισμός

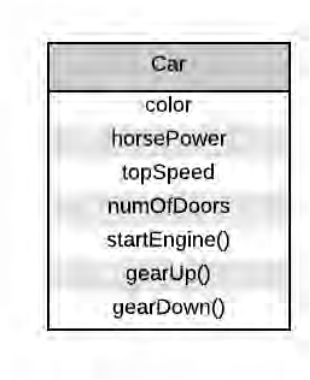
Πριν την υλοποίηση του συστήματος, θα πρέπει πρώτα να αναφερθούν μερικές πληροφορίες για την αντικειμενοστρέφεια και τα πεδία που είναι κοντινά της. Όταν οι άνθρωποι βλέπουν κάτι στο δρόμο, στο σπίτι ή ακόμα και στην εξοχή μπορούν κατά κάποιο τρόπο να το κατηγοριοποιήσουν σε κάποιο ανώτερο σύνολο από όμοια αντικείμενα.

Αυτή τη λογική δανείστηκαν οι προγραμματιστές και προχώρησαν στη δημιουργία των γλωσσών αντικειμενοστρεφούς προγραμματισμού. Αυτές,

είναι με τέτοιο τρόπο δομημένες έτσι ώστε να μπορεί κάποιος να εφαρμόσει αυτήν την κατηγοριοποίηση των αντικειμένων.

Τα βασικά στοιχεία του αντικειμενοστρεφούς προγραμματισμού που θα αναλύσουμε είναι:

- ***Κλάσεις***
- ***Αντικείμενα***
- ***Απόκρυψη πληροφορίας***
- ***Αφαιρετικότητα***



Σχήμα 3.3: *Κλάση Αυτοκίνητο*

Κλάσεις. Οι κλάσεις είναι η αρχή στον αντικειμενοστρεφή προγραμματισμό. Για παράδειγμα το αυτοκίνητο: το αυτοκίνητο έχει διάφορα *χαρακτηριστικά* και διάφορες *λειτουργίες*. Τα χαρακτηριστικά μπορεί να είναι η ιπποδύναμη, η τελική ταχύτητα, το χρώμα, το πλήθος των θηρών και άλλα. Επίσης κάποιες από τις λειτουργίες μπορεί να είναι η εκκίνηση της μηχανής, η αλλαγή ταχύτητας και άλλες. Αν το μοντελοποιηθεί αυτό σε κλάση, θα επέλθει το αποτέλεσμα που φαίνεται στο σχήμα 3.3. Επειδή όμως όλα τα αυτοκίνητα έχουν κάποια χαρακτηριστικά κοινά, η *κλάση Αυτοκίνητο* είναι σαν ένα πρωτότυπο για την παραγωγή πολλών *αντικειμένων* που στην περίπτωσή μας είναι αυτοκίνητα.

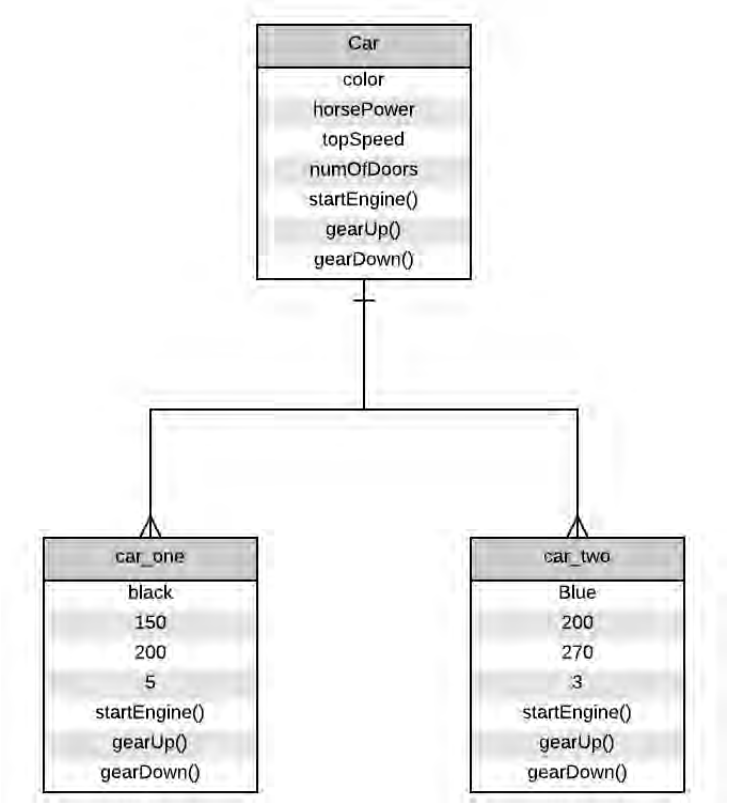
Γίνεται φανερό ότι σε μερικά πεδία της κλάσης υπάρχει μία παρένθενση στο τέλος, η οποία συμβολίζει ότι αυτά τα πεδία είναι οι λειτουργίες του αυτοκινήτου. Στον προγραμματισμό τα χαρακτηριστηκά τα λέμε *μεταβλητές (variables)* και τις λειτουργίες *συναρτήσεις (functions)* ή *μεθόδους (methods)*. Σε κώδικα το αποτέλεσμα θα είναι:

```
1  Class Car
2  {
3   private:
4    int gear;
5    bool engineOn;
6
7   public:
```

```
8     int horsePower;
9     int topSpeed;
10    char* color;
11    int numOfDoors;
12
13    void startEngine();
14    void gearUp();
15    void gearDown();
16
17  };
```

Θα εξηγηθεί παρακάτω η αιτία ύπαρξης κάποιων *μεταβλητών* που είναι *ιδιωτικές (private)*.

Αντικείμενα. Τα *αντικείμενα* παράγονται, όπως προαναφέρθηκε, από τις κλάσεις. Άρα ένα αντικείμενο ανήκει σε κάποια συγκεκριμένη κλάση, οπότε και τα χαρακτηριστικά που έχει η κλάση, έχουν πλέον τιμές όπως φαίνεται στο σχήμα 3.4. Παρατηρείται ότι από μία *κλάση* μπορούν να γεννηθούν πολλά *αντικείμενα*, με διαφορετικά χαρακτηριστηκά. Φαίνεται λοιπόν ότι η *κλάση* είναι σαν ένα πρωτότυπο για τη δημιουργία αντικειμένων. Για να δημιουργηθεί όμως ένα αντικείμενο πρέπει να κληθεί ο λεγόμενος *δημιουργός (constructor)*, δηλαδή μια βασική μέθοδος που πρέπει να έχουν όλες οι *κλάσεις*.



Σχήμα 3.4: *Αντικείμενα* της *κλάσης Αυτοκίνητο*.

Απόκρυψη πληροφορίας. Η *απόκρυψη πληροφορίας* ή αλλιώς

encapsulation είναι ένα σημαντικό στοιχείο της αντικειμενοστρέφειας. Όπως προειπώθηκε, στο παράδειγμα με τον κώδικα, έχουμε κάποιες *μεταβλητές* που είναι *ιδιωτικές*. Αυτό συμβαίνει γιατί ο προγραμματιστής που θα χρησιμοποιήσει την αντίστοιχη *κλάση*, δε χρειάζεται να γνωρίζει λεπτομέρειες για την υλοποίηση, αρκεί να έχει γενική γνώση των κινήσεών του..

Αφαιρετικότητα. Η *αφαιρετικότητα* ανάγεται από την *απόκρυψη πληροφορίας*, δηλαδή δίνει έμφαση στο τί είναι ένα *αντικείμενο* ή τί κάνει, παρά στο πώς παρουσιάζεται ή πώς δουλεύει. Είναι πολύ σημαντική για την κατασκευή προγραμμάτων, διότι μειώνει την πολυπλοκότητα όταν υπάρχουν εκτεταμένες γραμμές κώδικα.

3.4 Υλοποίηση και επεξήγηση του κώδικα

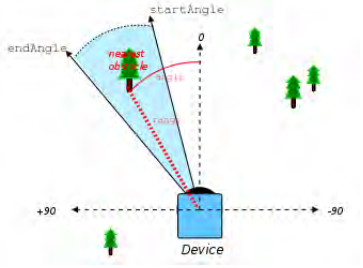
Σε αυτό το σημείο θα παρουσιαστούν τα κύρια κομμάτια κώδικα και γενικά πως υλοποιήθηκε το σύστημα. Έχουν μοντελοποιηθεί οι τρεις καταστάσεις, ως τρεις κλάσεις όπου η κάθε μία έχει κάποια χαρακτηριστικά και κάποιες μεθόδους. Οι τελευταίες παρουσιάζονται στο παράρτημα Α'. Η πρώτη κλάση που θα αναλυθεί είναι αυτή που υλοποιεί την *ακολούθηση*. Υλοποιείται σε δύο αρχεία Follow.cpp, Follow.h.

Στο αρχείο Follow.h φαίνεται αυτό που ονομάστηκε ως το πρωτότυπο της *κλάσης*. Άξιες αναφοράς είναι δύο γραμμές κώδικα το συγκεκριμένο αρχείο. Σε αυτές τις γραμμές γίνεται η δήλωση του *δημιουργού* και της βασικής συνάρτησης, όπου είναι υλοποιημένη όλη η λογική της κατάστασης.

```
1  Follow(ArRobot *robot, ArSonarDevice *sonar);
2  int run(int centerOfBB[2]);
```

./src/Follow.h

Στο αρχείο Follow.cpp υπάρχει η υλοποίηση των μεθόδων που παρουσιάστηκαν παραπάνω. Στο *δημιουργό* δεν υπάρχει κάτι αξιοσημείωτο, απλά αρχικοποιούνται κάποιες μεταβλητές. Η συνάρτηση *run()* που είναι η καρδιά της *κλάσης* Follow είναι αυτή που πρέπει να αναλυθεί. Στις πρώτες δύο γραμμές, δημιουργείται το αντικείμενο *move* το οποίο συντελεί στην πιο αφαιρετική όψη των κινήσεων του ρομπότ. Επίσης ορίζεται η μεταβλητή *range*, η οπία της αναθέτει την τιμή της απόστασης του κοντινότερου εμποδίου που βρίσκεται στην εμβέλια από τις -90°μέχρι τις 90°, όπως φαίνεται και στο σχήμα 3.5.



Σχήμα 3.5: Ένα παράδειγμα εντοπισμού εμποδίων με συσκευή εμβέλειας.

Έτσι πρωταρχική ευθύνη είναι να ελέγχθεί άν η απόσταση αυτή είναι μικρότερη από την απόσταση ασφαλείας που έχει οριστεί. Πιο επίσημα αν d είναι η απόσταση από το κοντινότερο εμπόδιο, και S η επιτρεπόμενη απόσταση ασφαλείας, τότε θα πρέπει να ισχύει ότι $d > S$ για να μπορέσει να κινηθεί το ρομπότ.

Ακολούθως ελέγχεται η ύπαρξη στόχου, που επιτυγχάνεται με τα δεδομένα τα οποία στέλνονται από το σύστημα που κάνει την αναγνώριση. Τα δεδομένα που λαμβάνει η συνάρτηση σαν όρισμα είναι απλά μια *μεταβλητή*. Ας την ορίσουμε x που μας δίνει την απόσταση του στόχου από το κέντρο της εικόνας. Στην περίπτωση μη ύπαρξης στόχου τότε το x θα έχει την τιμή -1, ενώ στην περίπτωση ύπαρξης στόχου, που σημαίνει $x \neq -1$, σύμφωνα με δύο συνθήκες μπορούν να δωθούν οι ανάλογες εντολές στο ρομπότ για να εκτελέσει τις αντίστοιχες κινήσεις. Δηλαδή αν οριστεί ως w το συνολικό πλάτος της εικόνας, τότε αν ισχύει ότι:

$$\frac{1}{3}w \leq x \leq \frac{2}{3}w \tag{3.1}$$

το ρομπότ θα κινηθεί μπροστά με μία ταχύτητα a. Αλλιώς αν:

$$\frac{1}{3}w > x \cup x > \frac{2}{3}w \tag{3.2}$$

τότε το ρομπότ θα κάνει μια περιστροφική κίνηση, όπου η περιστροφική ταχύτητα θα εξαρτάται από ένα βάρος W. Έτσι θα έχουμε:

$$\omega = W\omega_{max} \tag{3.3}$$

όπου το $W = \frac{2x-w}{w}$ και $W \exists [-1, 1]$. Η υλοποίηση σε κώδικα παρουσιάζεται παρακάτω.

```
1   // Target in line of sight
2   else if(centerOfBB[0] != -1)
3   {
4     // Target in the middle
5     if(WIDTH/3 <= centerOfBB[0] && centerOfBB[0] <= 2*WIDTH/3 && range
      > STOPDISTANCE)
6     {
7       move.forward(250);
8 //      printf("forward\n");
9     }
10    // Target is not in the middle
11    else if(centerOfBB[0] < WIDTH/3 || centerOfBB[0] > 2*WIDTH/3)
12    {
13      // This is a formula so the robot will turn a percentage of the
      max rotational velocity
14      float rotVelW = (2*centerOfBB[0] - WIDTH) / WIDTH;
15      move.turn(-rotVelW*maxRotVel);
16 //      printf("turn r=%f, 2*x=%d,-w=%d \n",rotVelW,2*centerOfBB[0],
      WIDTH);
17    }
18    return 1;
19  }
```

./src/Follow.cpp

Η κατάσταση *πήγαινε* υλοποιείται στα αρχεία GoTo.h kai GoTo.cpp. Στο πρώτο δεν είναι ανάγκη να δώθεί έμφαση σε κάτι συγκεκριμένο, είναι απλά το πρότυπο της *κλάσης* μας. Στο δεύτερο αρχείο, όπου και βρήσκεται η υλοποίηση, θα πρέπει να εξηγηθεί η συνάρτηση *run()* που είναι ο πυρήνας της *κλάσης*, όπως φάνηκε άλλωστε παραπάνω. Επίσης η υλοποίηση του *δημιουργού* δεν είναι καθόλου περίπλοκη γιατί η μόνη μας ευθύνη είναι η αρχηκοποίηση κάποιων *μεταβλητών*.

Ξεκινώντας την υλοποίηση της *run()*, δημιουργείται ένας πίνακας που περιέχει αλφαριθμητικά, τα οποία είναι τα ονόματα των συγκεκριμένων σημείων στο χάρτη που πρέπει να πάει το ρομπότ, όπως φαίνεται και στο σχήμα 3.6. Γενικά έχουμε δύο *μεταβλητές* που συμβάλλουν σε κάποιες συγκεκριμένες λειτουργίες, την *μεταβλητή* για να γνωρίζουμε το σημείο που θα σταλεί το ρομπότ μετά (goalCycle), και τη *μεταβλητή* που δείχνει το σχεδιασμό κάποιας διαδρομής για οποιοδήποτε σημείο (pathPlaned). Ο πρώτος έλεγχος που γίνεται αφορά την ύπαρξη στόχου και αν στην περίπτωση που δεν υπάρχει τότε θα προχωρήσω, αλλιώς εάν δεν υπάρχει στόχος τότε θα αλλάξουμε κατάσταση και θα προχωρήσουμε στην κατάσταση της *ακολούθησης*, όπως φαίνεται παρακάτω.

```
1  // Target aquired
2  if(currentBBCenter[0] != -1){
3    goalCycle = 0;
4    pathPlaned = 0;
5    return 1; //Time to follow target
6  }
```

./src/GoTo.cpp

Ο επόμενος έλεγχος που διεξάγεται είναι αν υπάρχει κάποιο σχέδιο διαδρομής για κάποιο από τα σημεία πάνω στο χάρτη. Αν δεν υπάρχει, τότε θα πρέπει να γίνει ο σχεδιασμός της διαδρομής. Η υλοποίηση γίνεται με τον αντίστοιχο κώδικα.

```
1    // Plan to pose
2    myPathTask->pathPlanToGoal(goals[goalCycle]);
3    cout << goals[goalCycle] << endl;
4    sleep(3);
```

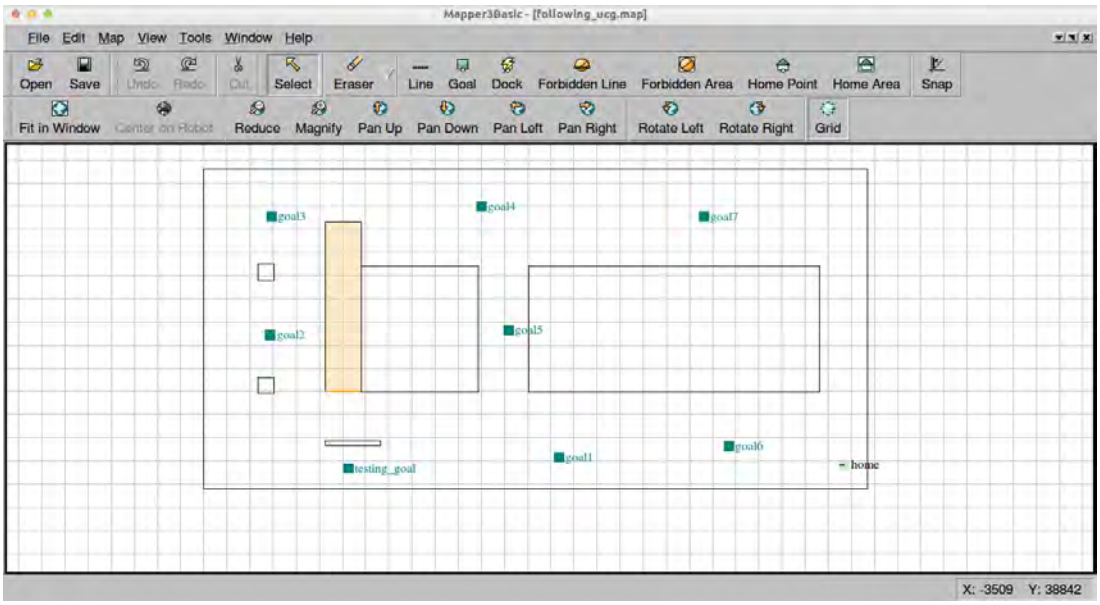
./src/GoTo.cpp

Άρα αφού θα σχεδιαστεί η διαδρομή θα πρέπει να αλλάξουμε τις *μεταβλητές* goalCycle και pathPlaned, δηλαδή να προχωρήσουμε στο επόμενο σημείο και να ορίσουμε οτι έχουμε κάποια διαδρομή. Τα επόμενα βήματα είναι πολύ απλά, αν τελικά φτάσουμε στον προορισμό μας, αλλάζουμε πάλι την emphμεταβλητή που μας ορίζει αν έχουμε διαδρομή ή οχι και δίνουμε τη σειρά στην κατάσταση αναζήτησης.

Τέλος θα αναλύθεί η κύρια συνάρτηση, δηλαδή η συνάρτηση που καλείται όταν τρέξουμε το πρόγραμμά μας. Στο αρχείο directRobotServer.cpp έχει γραφτεί ο κώδικας εκείνος που κάνει κάποιες βασικές αρχικοποιήσεις που υπάρχουν σχεδόν σε κάθε πρόγραμμα αυτού του τύπου. Τα κύρια σημεία που πρέπει να επισημανθούν σε αυτό το αρχείο είναι:

- Το αντικείμενο και τη μέθοδος για τον εντοπισμό θέσης.

- Το αντικείμενο για το σχεδιασμό διαδρομής.
- Τα τρία αντικείμενα των καταστάσεων.
- Την κύρια επανάληψη.



Σχήμα 3.6: Πρόγραμμα για την κατασκευή χαρτών.

Τα πρώτα δύο είναι πολύ εύκολο να υλοποιηθούν γιατί παρέχονται τέτοιες δυνατότητες από τη βιβλιοθήκη ARIA και φαίνονται στις παρακάτω γραμμές κώδικα. Ο αρχικός εντοπισμός θέσης γίνεται με την εντολή **locTask.localizeRobotAtHomeBlocking()**. Με την δημιουργία του αντικειμένου **locTask** ξεκινάει να τρέχει ένα *νήμα* του οποίου ο στόχος είναι να παρακολουθεί το σωστό εντοπισμό της θέσης του ρομπότ. Έτσι λειτουργεί και το αντικείμενο **pathTask**, δηλαδή όταν καλείται η μέθοδος **myPathTask->pathPlanToGoal(goals[goalCycle]);** ξεκινάει ένα *νήμα* που ελέγχει την πορεία του ρομπότ και αποφεύγει δυναμικά εμπόδια.

```
1    ArPathPlanningTask pathTask(&robot, &sonar, &map);
2
3    ArLog::log(ArLog::Normal, "Creating sonar localization task");
4    ArSonarLocalizationTask locTask(&robot, &sonar, &map);
```

./src/directRobotServer.cpp

Ακριβώς πριν ξεκινήσει η κύρια επανάληψη δημιουργούνται τα τρία *αντικείμενα* για τις τρεις καταστάσεις μας. Αφού δημιουργηθούν, είναι πλέον έτοιμη η εκκίνηση της κύριας επανάληψης. Αυτή περιλαμβάνει την εγκατάσταση επικοινωνίας με το σύστημα αναγνώρισης του στόχου, τη λήψη δεδομένων και τον έλεγχο των καταστάσεων.

Σε ψευδοκώδικα μπορούμε να το αναπαραστήσουμε ως εξής:

1. Λήψη δεδομένων.
2. Επιλογή κατάστασης.
3. Επεξεργασία δεδομένων.
4. Πήγαινε ξανά στο βήμα 1.

Η κάθε κατάσταση επιστρέφει έναν ακέραιο που σηματοδοτεί ποια θα είναι η επόμενη κατάσταση. Έτσι, με κάθε δεδομένο γίνεται αποδεκτό, το επεξεργάζεται και η αντίστοιχη κατάσταση.

Κεφάλαιο 4

Περιγραφή αλγορίθμων

4.1 Αλγόριθμοι εντοπισμού θέσης

Σε αυτό το κεφάλαιο θα παρουσιαστεί η περιγραφή ενός αλγορίθμου που χρησιμοποιείται για τον εντοπισμό της θέσης του ρομπότ στο χώρο. Γενικά έχουν διεξαχθεί εκτενείς έρευνες σχετικά με αυτόν τον τομέα, γιατί το να έχει το ρομπότ γνώση της θέσης του στο χώρο και γενικότερα πληροφοριών για το περιβάλλον του, είναι μεγάλο πλεονέκτημα για οποιαδήποτε εργασία θα πραγματοποιήσει μετά. Ο εντοπισμός της θέσης διακρίνεται σε δύο κύρια προβλήματα. Το ένα είναι η παρακολούθηση της θέσης του κατά τη διάρκεια μιας κίνησης και το άλλο είναι ο εντοπισμός της θέσης κατά την εκκίνηση του ρομπότ. Οι πρώτες έρευνες που έγιναν βασίστηκαν στο πρώτο πρόβλημα. Μετά το 1990 άρχισαν να εμφανίζονται έρευνες με αλγορίθμους που λύνουν και τα δύο προβλήματα. Η ιδέα για την εκτίμηση της θέσης μέσω πιθανότητας, έχει τις ρίζες της στα φίλτρα Kalman.[21, 32]

4.2 Αλγόριθμος εντοπισμού του Markov

Από τους πρώτους αλγορίθμους που αναπτύχθηκαν είναι του Markov.[27, 30, 14, 33] Πάνω σε αυτόν είναι βασισμένος και ο αλγόριθμος του Monte Carlo.[18] Έτσι θα γίνει μία αναφορά πρώτα στον αλγόριθμο του Markov για να γίνει καλύτερα κατανοητός καλύτερα ο MCL. Η βασική ιδέα του αλγορίθμου είναι να υπολογιστεί μια κατανομή πιθανότητας ανάμεσα σε όλες τις θέσεις του περιβάλλοντος. Ορίζουμε ως $l = \langle x, y, \theta \rangle$ τη θέση του ρομπότ στον χώρο, όπου το x και y είναι οι συντεταγμένες του ρομπότ σε ένα καρτεσιανό επίπεδο και το θ είναι η γωνία που είναι στραμμένο. Όπως ειπώθηκε θα χρειαστεί μια κατανομή η οποία συμβολίζεται ως $Bel(l)$ και δείχνει την *εμπιστοσύνη* για την θέση του ρομπότ. Η κατανομή θα ξεκινάει με κέντρο την αρχική θέση του ρομπότ, αν είναι γνωστή. Από την άλλη, θα είναι ομοιόμορφα κατανεμημένη αν η θέση είναι άγνωστη, για να φαίνεται ότι υπάρχει αυτή η αβεβαιότητα της θέσης. Όσο το ρομπότ θα λειτουργεί, αυτή η κατανομή θα βελτιώνεται.

Ο αλγόριθμος αποτελείται από δύο τρόπους για την βελτίωση της *εμπιστοσύνης*. Ο ένας βασίζεται στην κίνηση που κάνει το ρομπότ και ο άλλος βελτιώνει

την *εμπιστοσύνη* με τις πληροφορίες που συλλέγουν οι αισθητήρες.

Η κίνηση του ρομπότ μοντελοποιείται με την πιθανότητα υπό συνθήκη $P(l|l', \alpha)$, η οποία δείχνει την πιθανότητα του ρομπότ να καταλήξει στη θέση l αν από τη θέση l' εκτελέσει την κίνηση α. Έτσι, μετά την κίνηση η *εμπιστοσύνη* $Bel(l)$ ενημερώνεται σύμφωνα με την εξίσωση που συναντάμε στις αλυσίδες Markov[15]:

$$Bel(l) \longleftarrow \int P(l|l', \alpha)Bel(l')dl' \tag{4.1}$$

Με τον όρο $P(l|l', \alpha)$, μοντελοποιείται η κινηματική του ρομπότ. Η πιθανότητα υπάρχει γιατί θεωρείται ότι θα έχουμε λάθη στην οδομετρία του ρομπότ. Επίσης θεωρείται ότι τα λάθη ακολουθούν κανονική κατανομή, όπως επισημαίνει ο Burgard *et al*[18].

Οι πληροφορίες από τους αισθητήρες ενσωματώνονται στον κανόνα του Bayes. Η πληροφορία ενός αισθητήρα συμβολίζεται με το αγγλικό γράμμα s. Η πιθανότητα $P(s|l)$ δείχνει πόσο πιθανό είναι το ρομπότ να βρίσκεται στη θέση l αν πάρουμε σαν δεδομένο την πληροφορία του αισθητήρα s. Έτσι η *εμπιστοσύνη* ενημερώνεται σύμφωνα με τον παρακάτω κανόνα:

$$Bel(l) \longleftarrow \alpha P(s|l)Bel(l) \tag{4.2}$$

Όπου ο όρος α είναι ένας παράγοντας κανονικοποίησης.

Στην πραγματικότητα για να εφαρμοστούν οι παραπάνω μέθοδοι θα πρέπει το περιβάλλον να είναι *Μαρκοβιανό*, δηλαδή η πληροφορίες που δέχεται το ρομπότ από τους αισθητήρες να είναι ανεξάρτητες από τις πληροφορίες που θα έχουμε αν ξέρουμε την πραγματική θέση του ρομπότ. Βέβαια υπάρχουν έρευνες[19] που λύνουν το πρόβλημα με *μη-Μαρκοβιανά* περιβάλλοντα. Ο Burgard αναφέρει πως αφού μπορούν να εφαρμοστούν οι μέθοδοι του Fox *et al*[19] στον αλγόριθμο MCL,[18] τότε μπορεί να θεωρηθεί ότι όλα τα περιβάλλοντα είναι *Μαρκοβιανά*.

4.3 Αλγόριθμος Monte Carlo

Ο αλγόριθμος εντοπισμού θέσης του Monte Carlo(MCL) είναι μια επέκταση του αλγορίθμου εντοπισμού θέσης του Markov. Σύμφωνα με τους Burgard *et al*[18] είναι μια εκδοχή ενός αλγορίθμου που βασίζεται στη δειγματοληψία. Και μετά από μια προεργασία στη εκ των υστέρων δειγματοληψία (sampling/importance re-sampling) ή αλλιως SIR[29]. Ο αλγόριθμος είναι γνωστός και ως ''bootstrap filter''[22], ''Monte-Carlo filter''[25], ''Condensation algorithm''[23], ή ''survival of the fittest algorithm''[24]. Γενικά όλοι αυτοί οι αλγόριθμοι είναι τα λεγόμενα *φίλτρα σωματιδίων (particle filters)*. Περισσότερες πληροφορίες μπορούν να βρεθούν στην παρακάτω αναφορά [16].

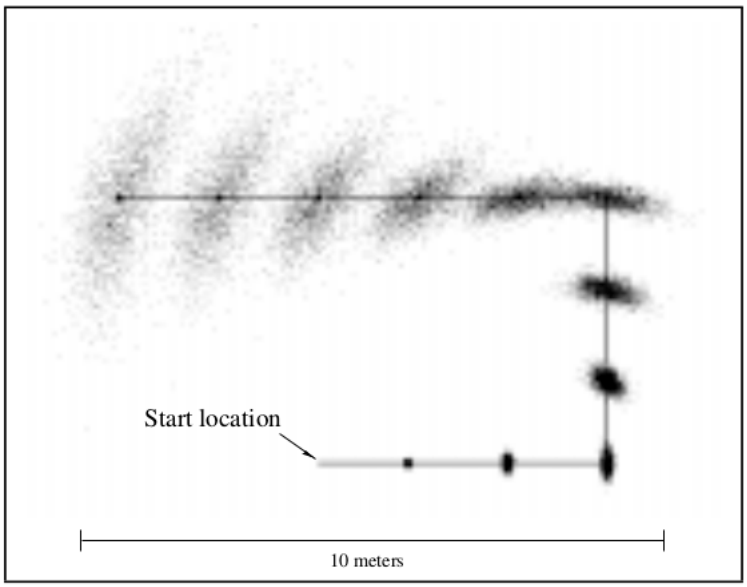
Η βασική ιδέα με αυτόν τον αλγόριθμο είναι να απεικονιστεί η εκ των υστέρων *εμπιστοσύνη* $Bel(l)$ σαν ένα σύνολο N τυχαίων δειγμάτων ή αλλιώς *σωματιδίων (particles)* $S = s_i|i = 1..N$. Το σύνολο αυτό είναι μια διακριτή προσέγγιση μιας κατανομής πιθανότητας. Τα δείγματα ορίζονται ως:

$$\langle\langle x, y, \theta\rangle, p\rangle \tag{4.3}$$

Όπου ο όρος $\langle x, y, \theta\rangle$ είναι η θέση του ρομπότ και το $p \geq 0$ είναι μια πιθανότητα που χρησιμοποιείται σαν βάρος τέτοια ώστε $\sum_{n=1}^{N} p_n = 1$.

Όπως και με τον αλγόριθμο του Markov ο MCL αποτελείται απο δύο φάσεις:

Κίνηση του ρομπότ. Όταν το ρομπότ κινηθεί ο MCL παράγει ένα καινούριο σύνολο από N δείγματα που προσεγγίζουν τη θέση του ρομπότ. Κάθε νέο δείγμα παράγεται από το σύνολο δειγμάτων που υπάρχει ήδη. Θεωρείται l' η θέση από το δείγμα που επιλέγουμε από τα ήδη υπάρχοντα, η θέση l του καινούριου δείγματος παράγεται από την πιθανότητα $P(l|l', \alpha)$ όπου το α είναι η κίνηση που εκτέλεσε το ρομπότ έτσι όπως παρατηρήθηκε. Τέλος η τιμή p του καινούριου δείγματος θα είναι N^{-1}. Στο σχήμα 4.1 βλέπουμε πώς διασπάται η



Σχήμα 4.1: Προσέγγιση της θέσης μέσο δειγματοληψίας

κατανομή σύμφωνα με την κίνηση του ρομπότ. Αυτή η διασπορά είναι που φανερώνει την αβεβαιότητα κατά τη διάρκεια της κίνησης του ρομπότ.

Η πληροφορία των αισθητήρων εφαρμόζεται κάνοντας μία αλλαγή στα βάρη κάθε δείγματος, με τον ίδιο τρόπο που γίνεται και στον αλγόριθμο του Markov. Πιο συγκεκριμένα θεωρείται $\langle l, p\rangle$ ένα δείγμα, τότε:

$$p \longleftarrow \alpha P(s|l) \tag{4.4}$$

Όπου το s είναι η πληροφορία που παίρνουμε από τον αισθητήρα και α είναι μια σταθερά κανονικοποίησης τέτοια ώστε $\sum_{n=1}^{N} p_n = 1$. Η αλλαγή των βαρών έχει δύο βήματα: το πρώτο, που το βάρος p πολλαπλασιάζεται με την πιθανότητα $P(s|l)$ και το δεύτερο που κανονικοποιείται με την σταθερά α. Ένας αλγόριθμος που το πετυχαίνει σε χρόνο $O(N)$, μπορεί να βρεθεί στην έρευνα των Carpenter, Clifford, & Fernhead 1997[17].

Στην πραγματικότητα οι Burgard *et al*[18] βρήκαν χρήσιμο να προσθέσουν ένα μικρό πλήθος από τυχαία δείγματα μετά από κάθε εκτίμηση των βαρών. Επειδή ο αλγόριθμος MCL δουλεύει με πεπερασμένο σύνολο δειγμάτων, μπορεί να μην παραχθεί κάποιο δείγμα που να είναι κοντά στην πραγματική θέση του ρομπότ. Με τον τρόπο αυτό, με την προσθήκη των τυχαίων δειγμάτων το ρομπότ μπορεί να ξαναβρεί την θέση του, σε περίπτωση που χαθεί.

Κεφάλαιο 5

Συμπεράσματα

Με την περαίωση αυτής της εργασίας, λογικό επακόλουθο είναι να διεξαχθούν κάποια συμπεράσματα. Το ότι μπορούμε με απλές διαδικασίες να υλοποιήσουμε συστήματα τέτοιου βεληνεκούς είναι ένα ενθαρρυντικό στοιχείο που αποδεικνύει την πορεία του συγκεκριμένου πεδίου, δηλαδή της ρομποτικής και της τεχνητής νοημοσύνης.

Το κυριότερο που αποκομίσαμε από την ολοκλήρωση της συγκεκριμένης εργασίας ήταν εμπειρία για το μέλλον. Η υλοποίηση ενός τέτοιου συστήματος, όχι μόνο συνετέλεσε στην κατανόηση της λειτουργίας και της δομής μιας ρομποτικής εφαρμογής, αλλά μας υπέδειξε και τον τρόπο σύνδεσης διαφορετικών συστημάτων με αποτελεσματικό και αποδοτικό τρόπο. Καίριας σημασίας θεωρείται επίσης το γεγονός ότι το σύστημα δουλεύει σε πραγματικό χρόνο, το οποίο είναι δύσκολο να επιτευχθεί αν αναλογιστεί κανείς το χρόνο που απαιτούν όλοι οι υπολογισμοί.

Εν κατακλείδι, θα παρουσιαστούν οι διάφορες βελτιώσεις, καθώς επίσης και κάποια μελλοντική δουλειά που μπορεί να γίνει και πάνω στο συγκεκριμένο πεδίο αλλά και σε άλλα θέματα που μπορούν να υπάρξουν γύρω από το PeopleBot[7] για ανάπτυξη.

- Καταρχήν το επόμενο βήμα πρέπει να είναι η βελτίωση του συστήματος με εισαγωγή κάμερας βάθους.
- Μια παραλλαγή του συστήματος θα ήταν το ρομπότ να εντοπίζει τη θέση του εντοπίζοντας γνωστά σημεία του χώρου μέσω κάμερας.
- Βελτίωση στις καταστάσεις, με προσθήκη νέων ή βελτίωση των παλαιών.
- Ίσως θα μπορούσε να επιδιωχθεί πιο περίπλοκος σκοπός, με παραπάνω από ένα ρομπότ, δηλαδή πλέον να έχει στόχο το ένα ρομπότ ένα άλλο ρομπότ.
- Καλό θα ήταν όλες οι δουλειές που γίνονται πάνω στο PeopleBot να καταγράφονται σε μια ιστοσελίδα τύπου ''wiki''.

Παράρτημα Α΄

Κώδικας

```
#include "Aria.h"

class Follow
{
private:
  ArRobot *myRobot;
  ArSonarDevice *mySonar;
  static const float WIDTH=640;
  static const int STOPDISTANCE=700;
  int maxRotVel;
public:
  Follow(ArRobot *robot, ArSonarDevice *sonar);
  int run(int centerOfBB[2]);
};
```

src/Follow.h

```
#include "Follow.h"
#include "Movement.h"
#include<stdio.h>
Follow::Follow(ArRobot *robot,ArSonarDevice *sonar)
{
  myRobot = robot;
  mySonar = sonar;
  maxRotVel = 72;
}

int Follow::run(int centerOfBB[2])
{
  double range = mySonar->currentReadingPolar(-90, 90) - myRobot->
    getRobotRadius();
  Movement move  = Movement(myRobot);

  // Not safe
  if(range < STOPDISTANCE)
  {
    move.stop();
    return 2;
  }
  // Target in line of sight
```

```
  else if(centerOfBB[0] != -1)
  {
    // Target in the middle
    if(WIDTH/3 <= centerOfBB[0] && centerOfBB[0] <= 2*WIDTH/3 && range
   > STOPDISTANCE)
    {
      move.forward(250);
//       printf("forward\n");
    }
    // Target is not in the middle
    else if(centerOfBB[0] < WIDTH/3 || centerOfBB[0] > 2*WIDTH/3)
    {
      // This is a formula so the robot will turn a percentage of the
   max rotational velocity
      float rotVelW = (2*centerOfBB[0] - WIDTH) / WIDTH;
      move.turn(-rotVelW*maxRotVel);
//       printf("turn r=%f, 2*x=%d,-w=%d \n",rotVelW,2*centerOfBB[0],
   WIDTH);
    }
    return 1;
  }
  // We have no target
  else
  {
    move.stop();
    return 2;
  }
}
```

src/Follow.cpp

```
#include"Aria.h"
#include"Arnl.h"
#include<iostream>
using namespace std;

class GoTo{
private:
  ArRobot *myRobot;
  ArMap *myMap;
  ArPathPlanningTask *myPathTask;
  ArPose myGoal;
  int pathPlaned;
  //char goals[7][7];
  int goalCycle;

public:
  //Constructor
  GoTo(ArRobot*, ArPathPlanningTask*, ArMap*);

  //Going to goal
  int run(int[]);
};
```

src/GoTo.h

```
#include "GoTo.h"

GoTo::GoTo(ArRobot *robot, ArPathPlanningTask *pathTask, ArMap *map){

  myRobot = robot;

  myMap = map;

  // Exits if there is no map
  if(myMap->getFileName() == NULL || strlen(myMap->getFileName()) == 0)
    {
     ArLog::log(ArLog::Normal,"No map");
     Aria::exit(1);
  }

  // Path task
  myPathTask = pathTask;

  // Don't have a plan yet
  pathPlaned = 0;

  // Goal Cycle
  goalCycle = 0;

}

// The main method
int GoTo::run(int currentBBCenter[2]){

  // Goal table
  char goals[7][7] = {"goal1","goal2","goal3","goal4","goal5","goal6","
    goal7"};

  // Target aquired
  if(currentBBCenter[0] != -1){
    goalCycle = 0;
    pathPlaned = 0;
    return 1; //Time to follow target
  }

  // Don't have a plan
  if(!pathPlaned){

    myRobot->clearDirectMotion();

    // Plan to pose
    myPathTask->pathPlanToGoal(goals[goalCycle]);
    cout << goals[goalCycle] << endl;
    sleep(3);

    // Now i have a plan
    pathPlaned = 1;

    // Next time next goal
    goalCycle++;
```

```
54
55      if(goalCycle > 6)
56      {
57        goalCycle = 0;
58      }
59
60      // Flow stays in GoTo
61      return 3;
62    }
63
64    // Reached target location give the flow to Search
65    if(myPathTask->getState() == ArPathPlanningTask::REACHED_GOAL){
66      // Plan completed
67      pathPlaned = 0;
68
69      return 2;// Search takes over each time the robot reaches a goal
        point
70    }
71
72    // If something goes wrong search takes over
73    // TODO maybe there is a better thing to do
74    if(myPathTask->getState() == ArPathPlanningTask::FAILED_PLAN){
75      pathPlaned = 0;
76      return 2;
77    }
78
79    // Moving to goal keep flow until goal reached
80    else{
81      return 3;
82    }
83  }
```

src/GoTo.cpp

```
1  #include "Aria.h"
2
3  class Movement
4  {
5    private:
6      ArRobot *myRobot;
7    public:
8      Movement(ArRobot *robot);
9      void forward(int velocity);
10     void turn(float rotVelocity);
11     void stop();
12 };
```

src/Movement.h

```
1  #include "Movement.h"
2  Movement::Movement(ArRobot *robot)
3  {
4    myRobot = robot;
5  }
6
7  void Movement::forward(int velocity)
```

```
{
  myRobot->lock();
  myRobot->setRotVel(0);
  myRobot->setVel(velocity);
  myRobot->unlock();
}

void Movement::turn(float rotVelocity)
{
  myRobot->lock();
  myRobot->setVel(0);
  myRobot->setRotVel(rotVelocity);
  myRobot->unlock();
}

void Movement::stop()
{
  myRobot->lock();
        myRobot->stop();
        myRobot->unlock();
}
```

src/Movement.cpp

```
#include "Aria.h"
#include "ArNetworking.h"
#include "Arnl.h"
#include "ArSonarLocalizationTask.h"
#include "ServerSocket.h"
#include "SocketException.h"
#include <string>
#include <iostream>
#include "Follow.h"
#include "GoTo.h"
#include "Search.h"


using namespace std;

#define WIDTH 640
#define HEIGHT 480
#define STOP_DISTANCE 700

int main ( int argc, char *argv[] )
{
  //cout << "running....\n";
  try{
    // Create the socket
    ServerSocket server ( 30000 );

    // Initialize Aria and Arnl global information
    Aria::init();
    Arnl::init();

    // The robot object
    ArRobot robot;
```

```
33
34      // Parse command line arguments
35      ArArgumentParser parser(&argc, argv);
36
37      // Set up our robot connector
38      ArRobotConnector robotConnector(&parser, &robot);
39
40      // Connect to the robot
41      if(!robotConnector.connectRobot())
42      {
43        ArLog::log(ArLog::Normal, "Error: could not connect to robot...\
    nexiting...");
44        Aria::exit(3);
45      }
46
47
48      // Set up where to look for files
49      // The default directory should be the projects' root directory
50      // addDirectories() appends "src" directory
51      char fileDir[1024];
52      ArUtil::addDirectories(fileDir, sizeof(fileDir), "../", "src");
53
54      // Add a section to the configuration to change ArLog parameters
55      ArLog::addToConfig(Aria::getConfig());
56
57      // Our mobile server for debugging
58      ArServerBase moServer;
59
60      // Set up our simpleOpener, used to set up the mobile server
61      ArServerSimpleOpener simpleOpener(&parser);
62
63      // Load default arguments for this computer (from /etc/Aria.args,
    environment variables
64      // and other places)
65      parser.loadDefaultArguments();
66
67      // Parse arguments
68      if(!Aria::parseArgs() || !parser.checkHelpAndWarnUnparsed())
69      {
70        ArLog::log(ArLog::Normal, "Exiting due to parsing error...");
71        Aria::exit(1);
72      }
73
74      // This causes Aria::exit(9) to be called if the robot unexpectedly
75      // disconects
76      ArGlobalFunctor1<int> shutdownFunctor(&Aria::exit, 9);
77      robot.addDisconnectOnErrorCB(&shutdownFunctor);
78
79      // Create an ArSonarDevice object (ArRangeDevice subclass) and
80      // connect it to the robot
81      ArSonarDevice sonar;
82      robot.addRangeDevice(&sonar);
83
84      // Start the robot thread
85      robot.runAsync(true);
86
```

```
87
88
89      /*** Create and set up map object ***/
90      // Set up the map object, this will look for files in the src
        directory
91      // (unless the file name starts with a '/', '\' or '.'
92      // You can take out the 'fileDir' argument to look in the program's
         current dir
93      // instead.
94      // When a configuration file is loaded into ArConfig later, if it
        specifies a
95      // map file, then that file will be loaded as the map
96      ArMap map(fileDir);
97
98
99
100     /*** Create localization and path planning threads ***/
101
102
103     ArPathPlanningTask pathTask(&robot, &sonar, &map);
104
105     ArLog::log(ArLog::Normal, "Creating sonar localization task");
106     ArSonarLocalizationTask locTask(&robot, &sonar, &map);
107
108
109
110
111
112     /*** Start the mobile server ***/
113
114     // Open the mobile server
115     if(!simpleOpener.open(&moServer, fileDir, 240))
116     {
117       ArLog::log(ArLog::Normal, "Error: Could not open mobile server.")
        ;
118       Aria::exit(2);
119     }
120
121     /* Create various services that provide network
122      * acces to clients (such as MobileEyes),
123      * as well as add various additional features
124      * to ARNL
125      */
126
127
128     /* Add additional range devices to the robot and
129      * path planning task (so it avoids obstacles
130      * detected by these devices)
131      */
132
133     robot.lock();
134     // Add bumpers range device to robot and path planning task
135     ArBumpers bumpers;
136     robot.addRangeDevice(&bumpers);
137     pathTask.addRangeDevice(&bumpers, ArPathPlanningTask::CURRENT);
138
```

```
139     // Add range device which uses forbidden regions given in the map
140     // to give virtual range device readings to ARNL
141     ArForbiddenRangeDevice forbidden(&map);
142     robot.addRangeDevice(&forbidden);
143     pathTask.addRangeDevice(&forbidden, ArPathPlanningTask::CURRENT);
144
145     robot.unlock();
146
147     // Action to slow down robot when localization score drops but not
    lost.
148     ArActionSlowDownWhenNotCertain actionSlowDown(&locTask);
149     pathTask.getPathPlanActionGroup()->addAction(&actionSlowDown, 140);
150
151     // Action to stop the robot when localization is "lost" (score too
    low)
152     ArActionLost actionLostPath(&locTask, &pathTask);
153     pathTask.getPathPlanActionGroup()->addAction(&actionLostPath, 150);
154
155
156 //    // Service to provide drawings of data in the map display
157 //    ArServerInfoDrawings drawings(&moServer);
158 //    drawings.addRobotsRangeDevices(&robot);
159 //
160 //    /* Draw a box around the local path planning area use this
161 //     * (You can enable this particular drawing from custom commands
162 //     * which is set up down below in ArServerInfoPath)
163 //     */
164 //    ArDrawingData drawingDataP("polyLine", ArColor(200,200,200), 1,
    75);
165 //    ArFunctor2C<ArPathPlanningTask, ArServerClient *, ArNetPacket *>
    drawingFunctorP(&pathTask, &ArPathPlanningTask::drawSearchRectangle
    );
166 //    drawings.addDrawing(&drawingDataP, "Local Plan Area", &
    drawingFunctorP);
167 //
168 //    // "Custom" commands. You can add your own custom commands here,
169 //    // they will be available in MobileEyes through Robot tools
170 //    ArServerHandlerCommands commands(&moServer);
171
172     // These provide various kinds of information to the client
173     ArServerInfoRobot serverInfoRobot(&moServer, &robot);
174     ArServerInfoSensor serverInfoSensor(&moServer, &robot);
175     ArServerInfoPath serverInfoPath(&moServer, &robot, &pathTask);
176 //    serverInfoPath.addSearchRectangleDrawing(&drawings);
177 //    serverInfoPath.addControlCommands(&commands);
178
179     // Provide localization info and allows the client (MobileEyes) to
    relocalize at a given
180     // pose
181     ArServerInfoLocalization serverInfoLocalization(&moServer, &robot,
    &locTask);
182     ArServerHandlerLocalization serverLocHandler(&moServer, &robot, &
    locTask, true, false);
183
184     // Provide the map to the client (and related controls)
185     ArServerHandlerMap serverMap(&moServer, &map);
```

```
186
187
188     /* Set up the possible modes for remote control from a client such
        as
189       * MobileEyes
190       */
191
192     // Mode to go to a goal or other specific point
193     ArServerModeGoto modeGoto(&moServer, &robot, &pathTask, &map,
        locTask.getRobotHome(), locTask.getRobotHomeCallback());
194
195     // Mode to stop and remain stopped
196     ArServerModeStop modeStop(&moServer, &robot);
197
198     // Make Stop mode the default
199     modeStop.addAsDefaultMode();
200
201
202     // Create a pose storage class, this will let the program keep
203     // track of where the robot is between runs
204     ArPoseStorage poseStorage(&robot);
205     if(poseStorage.restorePose("robotPose"))
206     {
207       serverLocHandler.localizeToPose(robot.getEncoderPose());
208     }
209
210
211     /*** Load configuration values, map, and begin! ***/
212
213     // When parsing the configuration file, also look at the program's
        command line options
214     // from the command-line argument parser as well as the
        configuration file
215     Aria::getConfig()->useArgumentParser(&parser);
216
217     // Read in parameter files
218     ArLog::log(ArLog::Normal, "Loading config file %s into ArConfig..."
        , Arnl::getTypicalParamFileName());
219     if(!Aria::getConfig()->parseFile(Arnl::getTypicalParamFileName()))
220     {
221       ArLog::log(ArLog::Normal, "Trouble loading configuration file,
        exiting...");
222       Aria::exit(5);
223     }
224
225     // Warn about unknown params
226     if(!simpleOpener.checkAndLog() || !parser.checkHelpAndWarnUnparsed
        ())
227     {
228       ArLog::log(ArLog::Normal, "Unknown parameters, exiting...");
229       Aria::exit(6);
230     }
231
232     // Print a log massage notifying user of the directory for map
        files
233     ArLog::log(ArLog::Normal, "");
```

```
234     ArLog::log(ArLog::Normal,
235         "Directory for maps and file serving: %s", fileDir);
236     ArLog::log(ArLog::Normal, "See the ARNL README.txt for more
        information");
237     ArLog::log(ArLog::Normal, "");
238
239
240     // Do an initial localization of the robot. It tries all the home
        points
241     // in the map, as well as the robot's current odometric position as
         possible
242     // places the robot is likely to be at startup. If successful it
        will
243     // also save the position it found to be the best localized
        position as the
244     // "Home" position, which can be obtained from the localization
        task (and is
245     // used by the "Go to home" network request)
246     locTask.localizeRobotAtHomeBlocking();
247
248     // Start the mobile server's thread
249     moServer.runAsync();
250
251
252     //Create the three states
253     robot.lock();
254     Follow follow = Follow(&robot,&sonar);
255     GoTo goTo(&robot,&pathTask,&map);
256     Search s(&robot,&sonar);
257     robot.unlock();
258
259     // Enable the motors
260     robot.enableMotors();
261
262
263     //Main loop
264     while (true){
265       //The socket to accept connection
266       ServerSocket new_sock;
267       server.accept ( new_sock );
268       int state = 1;    //1 = Follow, 2 = Search, 3 = GoTo
269       int lastPos[2];   //Storing last position of BB to search the
        target
270       int data[2];    //matrix with X,Y of BB
271       char cstate[3][10] = {"Following", "Searching", "Going to"};
272       int printCounter = 0;
273       try{
274         while ( true ){
275           //receive data from tld
276           new_sock >> data;
277
278
279           if(data[0] != -1)
280             lastPos[0] = data[0];
281
282           //Main logic
```

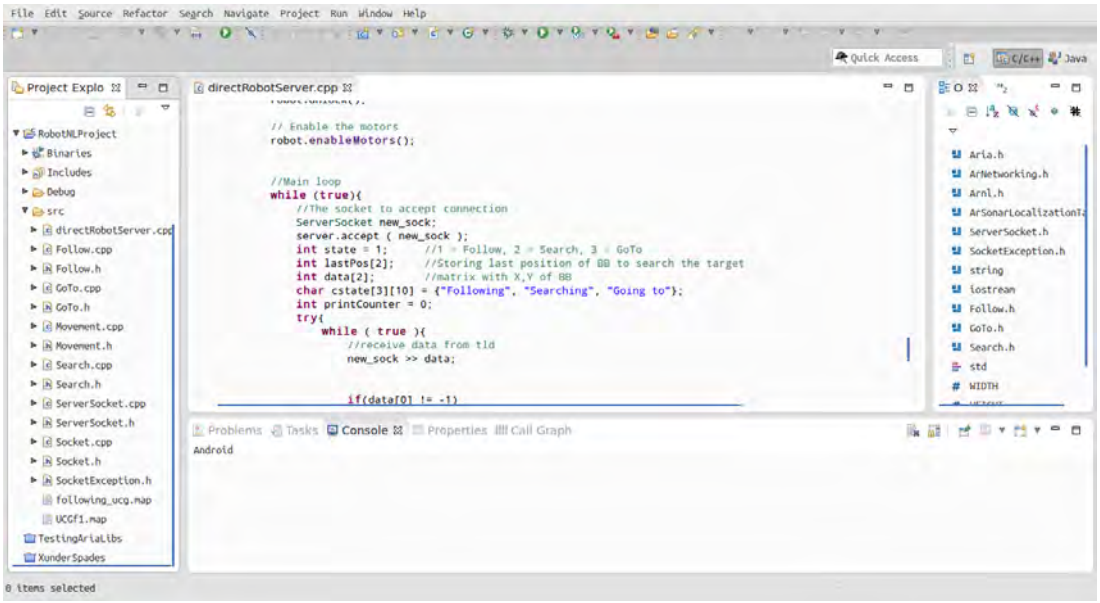
```
283           switch(state){
284
285           case 1:
286             state = follow.run(data);
287             break;
288           case 2:
289             state = s.seek(lastPos, data);
290             break;
291           case 3:
292             state = goTo.run(data);
293             break;
294           default:
295             state = 1;
296             break;
297           }
298           printCounter += 1;
299           if(printCounter > 10)
300           {
301             printCounter = 0;
302             std::cout << "
    ------------------------------------------------------------\n";
303             std::cout << "State: " << cstate[state-1] << std::endl;
304             std::cout << "Data: " << data[0] << ", " << data[1] << std
    ::endl;
305             std::cout << "Loc score: " << locTask.getLocalizationScore
    () << std::endl;
306             std::cout << "Odemetric pose: Th->" << robot.getTh() << " X
    ->" << robot.getX() << " Y->" << robot.getY() << std::endl;
307           }
308
309
310         }
311       }
312       catch ( SocketException& ) {
313         cout << "Lost Connection" << endl;
314         robot.lock();
315         robot.stop();
316         robot.unlock();
317       }
318     }
319   }
320   catch ( SocketException& e ){
321     std::cout << "Exception was caught:" << e.description() << "\
    nExiting.\n";
322   }
323
324   ArLog::log(ArLog::Normal, "RobotServer: Exiting.");
325   Aria::exit(0);
326 }
```

src/directRobotServer.cpp

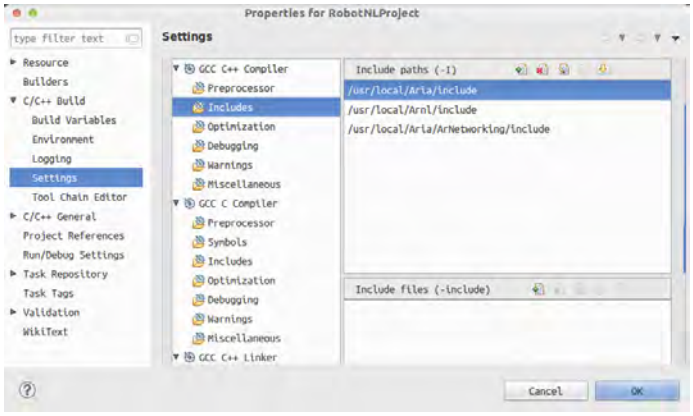
Παράρτημα Β΄

Σύνδεση της Aria με το Eclipse

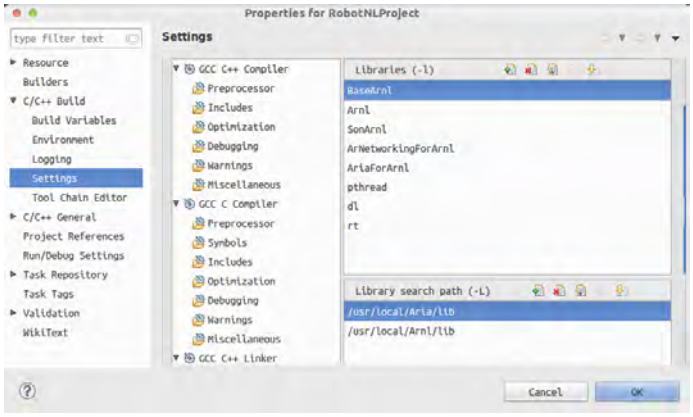
1. Δεξί κλίκ πάνω στον φάκελο του project στα αριστερά και επιλογή properties.
2. Επιλογή C/C++ Build και μετά Settings.
3. Στις επιλογές του GCC C++ Compiler προσθέτουμε τα αρχεία κεφαλίδων που είναι απαρέτειτα, όπως φένεται στο σχήμα Β'.2.
4. Στις επιλογές του GCC C++ Linker προσθέτουμε τις βιβλιοθήκες και τα απόλυτα μονοπάτια αυτών όπως φένεται στο σχήμα Β'.3



Σχήμα Β'.1: Το περιβάλλον Eclipse.



Σχήμα Β'.2: Τα αρχεία κεφαλίδων.



Σχήμα Β'.3: Οι βιβλιοθήκες.

Παράρτημα Γ΄

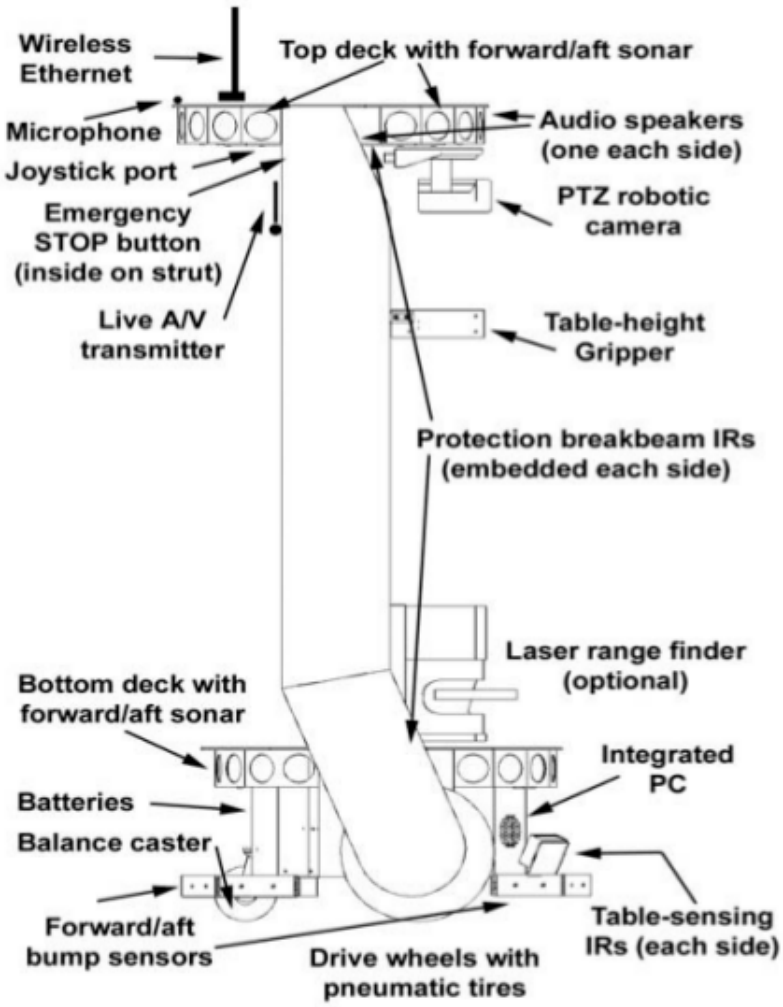
Χαρακτηριστικά του PeopleBot

Φυσικά χαρακτηριστικά	
Μήκος (cm)	47
Πλάτος (cm)	38
Ύψος (cm)	124
Βάρος (kg)	21
Φορτίο (kg)	11

Ισχύς	
Μπαταρίες 12VDC	3
Φόρτιση (watt ανά ώρα)	252
Χρόνος λειτουργίας (ώρες)	8 - 10
με Η/Υ (ώρες)	3 - 4
Χρόνος φόρτισης ώρες/μπατ.	6

Κινητικότητα	
Τροχοί	2 με αφρό
Πέλμα	τρακτεροτό
Διάμετρος (mm)	195.3
Πλάτος (mm)	50
Οδήγηση	διαφορική
Λόγος ταχύτητας	38.3:1
Μέγιστη ταχύτητα (mm/sec)	900
Μέγιστη ταχύτητα περιστροφική (deg/sec)	150
Διασχίσιμο κενό (mm)	50
Διασχίσιμη κλίση (ποσοστό)	11%
Επιτρεπτό έδαφος	ανάλογο με καροτσάκι

Πίνακας Γ'.1: Τα κύρια χαρακτηριστικά του PeopleBot



Σχήμα Γ'.1: Τα διάφορα κομμάτια του PeopleBot.

Βιβλιογραφία

[1] http://el.wikipedia.org/wiki/Microsoft_Windows.

[2] http://en.wikipedia.org/wiki/Linux.

[3] http://en.wikipedia.org/wiki/Robotics.

[4] http://er.jsc.nasa.gov/seh/Robotics/index.html.

[5] http://robots.mobilerobots.com/wiki/ARIA.

[6] http://www.mobilerobots.com/Mobile_Robots.aspx.

[7] http://www.mobilerobots.com/ResearchRobots/PeopleBot.aspx.

[8] http://www.shadowrobot.com/media/pictures.shtml.

[9] Imitation of Life: A History of the First Robots.

[10] Masters of Manufacturing: Joseph F. Engelberger.

[11] Adam and Currie. *The History of Robotics*. 1999.

[12] Isaac Asimov. 4 The Word I Invented. *Counting the Eons. Doubleday." Robotics has become a sufficiently well developed technology to warrant articles and books on its history and I have watched this in amazement, and in some disbelief, because I invented… the word*, 1983.

[13] Isaac Asimov and Robert Silverberg. *The Robot Chronicles*. 1995.

[14] Anthony R Cassandra, Leslie Pack Kaelbling, and James A Kurien. Acting under uncertainty: Discrete Bayesian models for mobile-robot navigation. In *Intelligent Robots and Systems' 96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, volume 2, pages 963--972. IEEE, 1996.

[15] Kai Lai Chung. *Markov chains with stationary transition probabilities*, volume 104. Springer New York, 1967.

[16] Arnaud Doucet, SJ Godsill, and C Andrieu. *On sequential simulation-based methods for Bayesian filtering*. Department of Engineering, University of Cambridge UK, 1998.

[17] P Fearnhead, J Carpenter, and P Clifford. An improved particle filter for non-linear problems. Technical report, Technical Report, 1983.

[18] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte carlo localization: Efficient position estimation for mobile robots. *AAAI/IAAI*, 1999:343--349, 1999.

[19] Dieter Fox, Wolfram Burgard, Sebastian Thrun, and Armin B Cremers. Position estimation for mobile robots in dynamic environments. In *AAAI/IAAI*, pages 983--988, 1998.

[20] Charles Coulston Gillispie and Frederic Lawrence Holmes. *Dictionary of scientific biography*, volume 12. Scribner, 1981.

[21] Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, volume 140, pages 107--113. IET, 1993.

[22] Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, volume 140, pages 107--113. IET, 1993.

[23] Michael Isard and Andrew Blake. Condensation---conditional density propagation for visual tracking. *International journal of computer vision*, 29(1):5--28, 1998.

[24] Keiji Kanazawa, Daphne Koller, and Stuart Russell. Stochastic simulation algorithms for dynamic probabilistic networks. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 346--351. Morgan Kaufmann Publishers Inc., 1995.

[25] Genshiro Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of computational and graphical statistics*, 5(1):1--25, 1996.

[26] Joseph Needham. *Science and Civilisation in China: Sections 8-18.-History of Scientific Thought*. Cambridge University Press, 1962.

[27] Illah Nourbakhsh, Rob Powers, and Stan Birchfield. DERVISH an office-navigating robot. *AI magazine*, 16(2):53, 1995.

[28] Mark E Rosheim. *Robot evolution: the development of anthrobotics*. Wiley. com, 1994.

[29] Donald B Rubin et al. Using the SIR algorithm to simulate posterior distributions. *Bayesian statistics*, 3:395--402, 1988.

[30] Reid Simmons and Sven Koenig. Probabilistic robot navigation in partially observable environments. In *IJCAI*, volume 95, pages 1080--1087, 1995.

[31] Balkeshwar Singh, N Sellappan, and P Kumaradhas. Evolution of Industrial Robots and their Applications.

[32] Randall Smith, Matthew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, pages 167--193. Springer, 1990.

[33] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Autonomous Robots*, 5(3-4):253--271, 1998.