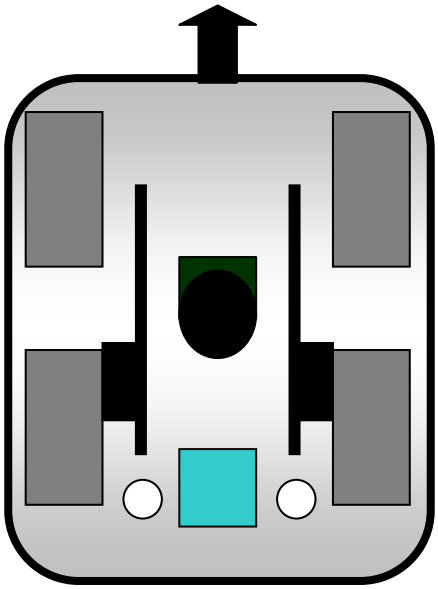
ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

Τμήμα Μηχανικών Παραγωγής & Διοίκησης .

Διπλωματική Εργασία

«Ανάπτυξη αναδρομικού αλγόριθμου πλοήγησης με την βοήθεια συσκευών υπερήχου και χαρτογράφησης δωματίου με το έντροχο ρομπότ ATRV-mini »



Διαμαντάκης Μιχάλης

Επιβλέπων : Τσουρβελούδης Νικόλαος

ΧΑΝΙΑ 2004

Ευχαριστίες

Στο σημείο αυτό θα ήθελα να ευχαριστήσω τους γονείς μου για την υλική και ηθική υποστήριξή τους τόσο κατά τη διάρκεια των σπουδών μου όσο και κατά την παρούσα διπλωματική εργασία. Επίσης θα ήθελα να εκφράσω τις ευχαριστίες μου στους καθηγητές μου Βαλαβάνη Κίμων και Τσουρβελούδη Νικόλαο για την πολύτιμη καθοδήγη και βοήθειά τους στην ολοκλήρωση της διπλωματικής εργασίας. Τέλος, ευχαριστώ όλους τους κοντινούς μου ανθρώπους για την ηθική υποστήριξή τους.

ΕΙΣΑΓΩΓΗ

Ένα από τα πλέον διαδεδομένα προβλήματα στο χώρο των έντροχων ρομπότ είναι η εύρεση αλγόριθμων πλοήγησης σε συνδυασμό με την ταυτόχρονη λειτουργία των αισθητήρων του ρομπότ για την επίτευξη κάποιου σκοπού [1]. Τα τελευταία χρόνια μεγάλες εταιρίες που ασχολούνται με τον προγραμματισμό των έντροχων ρομπότ, έχουν επικεντρωθεί στη χαρτογράφηση κλειστών χωρών αλλά και στη φύλαξη αυτών με την βοήθεια ρομπότ.

Σκοπός της διπλωματικής εργασίας είναι η ανάπτυξη αλγόριθμου κίνησης και παράλληλα η χρήση των αισθητήρων υπερήχων που διαθέτει το έντροχο ρομπότ ATRV-mini προκείμενου να χαρτογραφηθούν κλειστοί χώροι. [2].

Τα δομικά στοιχεία του συστήματος είναι το ρομπότ ATRV-mini, τέσσερις κλειστοί χώροι και ένας ηλεκτρονικός υπολογιστής. Από τους τέσσερις κλειστούς χώρους οι δυο περιέχουν εμπόδια και ο άλλος είναι κενός. Ο ηλεκτρονικός υπολογιστής επικοινωνεί με ασύρματο δίκτυο με το ρομπότ και χρησιμοποιείται για την γραφική αναπαράσταση των κλειστών χώρων.

Η χαρτογράφηση κλειστών χώρων είναι σημαντική για την αυτόματη παροχή πληροφοριών αρχιτεκτονικών δεδομένων, χωρίς την συμμετοχή του ανθρώπου. Η χαρτογράφηση γίνεται αυτοματοποιημένα, ανεξάρτητα από τον παράγοντα φως και παρέχονται πληροφορίες για χώρους όπου ο άνθρωπος δεν θα μπορούσε να έχει άμεση πρόσβαση [1].

Στο πρώτο κεφάλαιο θα παρουσιαστούν τα βασικά χαρακτηριστικά του ρομπότ. Θα γίνει παρουσίαση της λειτουργίας των αισθητήρων υπερήχων και των δυνατοτήτων του ρομπότ. Στο δεύτερο κεφάλαιο θα γίνει παρουσίαση του αλγορίθμου πλοήγησης, και πως αυτός εφαρμόστηκε σε επίπεδο προσομοίωσης. Στο τρίτο κεφάλαιο παρουσιάζεται η εφαρμογή του αλγόριθμου σε συνθήκες εργαστηρίου, η μορφή των αποτελεσμάτων και κάποια προβλήματα τα οποία αντιμετωπίστηκαν με κατάλληλους χειρισμούς. Στο κεφάλαιο αυτό παρουσιάζεται και η γραφική αναπαράσταση των

δωματίων που χαρτογραφεί το ρομπότ με την βοήθεια του προγράμματος Map Viewer.

Επιπρόσθετα, θα γίνει αναφορά στην δυνατότητα της περαιτέρω εξέλιξης της διπλωματικής, σε επιπλέον εφαρμογές τις οποίες μπορεί να εκτελέσει ο αλγόριθμος και θα οριστούν οι κώδικες που χρησιμοποιήθηκαν.

ΚΕΦΑΛΑΙΟ 1
Τεχνικά χαρακτηριστικά του ATRV-mini

1.1 Εισαγωγή

Το ρομπότ ATRV-mini όπως φαίνεται στην Εικόνα 1.1 είναι ένα έντροχο ρομπότ το οποίο έχει την δυνατότητα να πραγματοποιεί μια σειρά από λειτουργίες. Διαθέτει σύστημα κίνησης που αποτελείται από δυο ηλεκτροκινητήρες και τέσσερις τροχούς διαφορικής κίνησης. Ο κάθε ηλεκτροκινητήρας είναι συνδεδεμένος με ένα ζεύγος τροχών.

Το ρομπότ διαθέτει, επίσης, 24 αισθητήρες υπερήχων (sonar) οι οποίοι είναι διατεταγμένοι γύρω από το τροχοφόρο ρομπότ. Οι αισθητήρες έχουν ως σκοπό τον εντοπισμό εμποδίων που μπορεί να βρίσκονται σε καθορισμένο εύρος απόστασης γύρω από το ρομπότ. Οι αισθητήρες υπερήχου έχουν ευρύτατη χρήση, μερικά παραδείγματα αναφέρονται παρακάτω.

- Ιατρική: Απεικόνιση και διερεύνηση εσωτερικών οργάνων του ανθρώπινου σώματος
- Πολεμικό Ναυτικό: υποβρύχια

Ο τρόπος λειτουργίας των αισθητήρων υπέρηχων θα αναλυθεί διεξοδικά στη παράγραφο 1.3.

Έχει τοποθετηθεί κάμερα στο επάνω μέρος του ρομπότ προκειμένου να είναι δυνατή η λήψη φωτογραφιών σε όλη την διάρκεια της κίνησης του τροχοφόρου συστήματος. Η κάμερα κινείται περιστροφικά περί τον οριζόντιο και κατακόρυφο άξονα επιτυγχάνοντας ορατότητα προς όλες σχεδόν τις κατευθύνσεις.

Για την αποτελεσματικότερη λειτουργία του ρομπότ έχει ενσωματωθεί ηλεκτρονική πυξίδα προσανατολισμού και σύστημα παγκόσμιου προσδιορισμού θέσης (GPS -

Ground Positioning System). Φέρει, επίσης, δύο προφυλακτήρες εξοπλισμένους με αισθητήρες κρούσης (στο εμπρός και πίσω μέρος), διακόπτοντας τη λειτουργία των κινητήρων κατά τη σύγκρουση. Διαθέτει μία μπαταρία των 24 Volt για μικρές ταχύτητες κίνησης σε λεία και επίπεδη επιφάνεια, η οποία του παρέχει αυτονομία κινήσεων για 2-3 ώρες περίπου. Στον Πίνακα 1.1 παρουσιάζονται τα παραπάνω τεχνικά χαρακτηριστικά του ρομπότ.

Εικόνα 1.1: Αναπαράσταση ρομπότ ATRV-mini

ATRV-Mini	**SPECIFICATIONS**
Sonar	24 (6 front facing, 12 side facing, 6 rear facing; optional full array)
CPU's	Pentium based EBX computer system
Communications	Optional wireless radio Ethernet
Batteries	24V, 12A/hr
Run Time	2 to 4 hours terrain dependent
Motor	2 high 24VDC servo motors
Drive	4-wheel differential
Turn Radius	Zero (skid steer)
Translate Speed	1.5m/s 4.9'/s
Rotate Speed	250deg/s
Approach Angle	45deg
Decent Angle	45deg
Payload	9.1kg 20lbs
Height	45cm 18"
Length	60.5cm 23.8"
Width	53.3cm 21"
Weight	39kg 86lbs
Colors	Research Robot Red or special
Warranty	One year parts and labor
Accessories	Replacement Battery Vision Systems Wireless Communications Computerized Navigation Compass 12 Channel GPS Receiver Navigation Sensors Front and Rear Tactile Bumpers

Πίνακας 1.1: Τεχνικά χαρακτηριστικά του έντροχου ρομπότ ATRV-mini

1.2 Κίνηση του ρομπότ

Όπως προαναφέρθηκε το ATRV-mini διαθέτει 2 ηλεκτροκινητήρες κάθε ένας από τους οποίους κινεί ένα ζεύγος τροχών. Η κίνηση του ρομπότ είναι δυνατή με 2 τρόπους:

- με χειριστήριο, και
- με υπολογιστή που είναι συνδεδεμένος σε δίκτυο με το ρομπότ.

Στη περίπτωση κίνησης με χρήση χειριστηρίου οι εντολές κίνησης δίνονται από τον χειριστή εκείνη την στιγμή. Με τον τρόπο αυτό ο χειριστής έχει τον απόλυτο έλεγχο του ρομπότ.

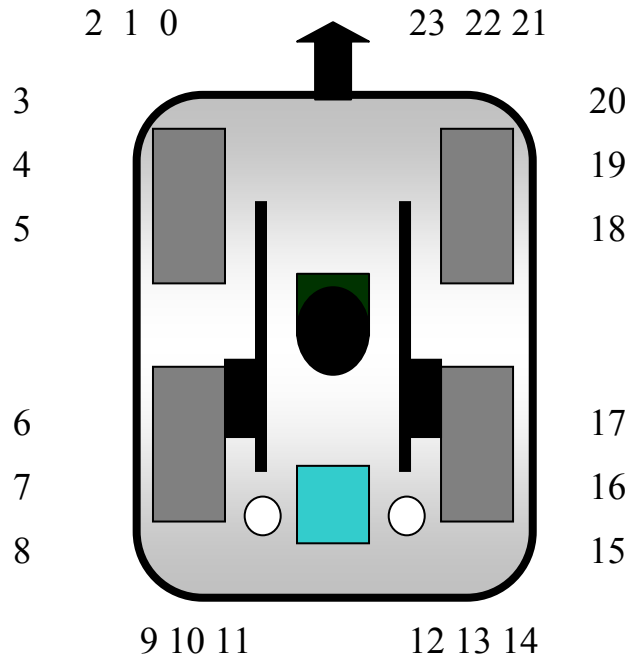
Στην περίπτωση κίνησης με χρήση ηλεκτρονικού υπολογιστή, η κίνηση του ρομπότ πραγματοποιείται μέσω λογισμικού. Στην περίπτωση αυτή το ρομπότ είναι συνδεδεμένο με υπολογιστή μέσω του οποίου δημιουργείται το κατάλληλο προς κίνηση λογισμικό και εν συνεχεία το ρομπότ εκτελεί τις κινήσεις. Σε αυτή την περίπτωση ο χειριστής οφείλει να ορίσει στο ρομπότ τις ακριβές κινήσεις που θα πρέπει να εκτελέσει.

Σε πολλές περιπτώσεις, η κίνηση του ρομπότ επιβάλλεται να πραγματοποιηθεί μέσω χρήσης υπολογιστή και όχι με την χρήση χειριστηρίου λόγο της αυτοματοποίησης που προσφέρει στην κίνηση του. Εξάλλου, η δημιουργία κατάλληλων για τη κίνηση του ρομπότ αλγορίθμων είναι πολύ σημαντική. Η περίπτωση λάθους περιορίζεται μόνο στο τρόπο λειτουργίας του προγράμματος από τον χρήστη. Επομένως, η κίνηση του ρομπότ πραγματοποιείται με μεγάλη θεωρητικά ακρίβεια. Η μείωση στην ακρίβεια της κίνησης του ρομπότ οφείλεται είτε στην ολισθηρότητα του δαπέδου είτε στην κακή κατάσταση των ελαστικών του ρομπότ. Οπότε θα πρέπει να πραγματοποιείται γενικός έλεγχος πριν την χρήση του.

1.3 Αισθητήρες Υπερήχων

Οι αισθητήρες υπερήχων είναι συσκευές οι οποίες έχουν σαν σκοπό τους την εύρεση κάποιου εμποδίου με την βοήθεια υπερήχων.

Στο ATRV-mini οι 24 αισθητήρες βρίσκονται περιμετρικά του ρομπότ όπως φαίνεται στην Εικόνα 1.2.



Εικόνα 1.2: Διάταξη αισθητήρων υπερήχου στο έντροχο ρομπότ ATRV-mini

και αριθμούνται από το 0 ως το 23. Η αρχή λειτουργίας βασίζεται στο γεγονός ότι ο ήχος έχει σταθερή ταχύτητα και ανακλάται όταν βρει κάποιο εμπόδιο. Όταν οι αισθητήρες στέλνουν σήματα (υπερήχους) αναμένεται η επιστροφή των σημάτων στους εκάστοτε δέκτες, εφόσον βρουν κάποιο εμπόδιο. Κατόπιν γίνεται ο υπολογισμός της απόστασης από τον εκάστοτε αισθητήρα. Η απόσταση του εμποδίου είναι η μισή της αρχικής απόστασης που υπολογίζεται αφού ο χρόνος εκπομπής και λήψης είναι διπλάσιος από αυτόν που πραγματικά χρειάζεται για να φτάσει ο ήχος από τον αισθητήρα στο εμπόδιο.

Κάθε αισθητήρας υπερήχων αποτελείται από ένα εσωτερικό δακτύλιο, ο οποίος περιέχει το ραβδωτό πιατίνι και την ειδική μεμβράνη-έλασμα που εκτελεί τις ταλαντώσεις. Το έλασμα αυτό προστατεύεται από μια καλύπτρα η οποία φέρει πόρους για τη διέλευση του κύματος.

Πρέπει να σημειωθεί ότι οι αποστάσεις εμφανίζονται είτε με την βοήθεια κατάλληλου λογισμικού μέσω ηλεκτρονικού υπολογιστή (όπως έγινε στην παρούσα διπλωματική) είτε στην οθόνη που υπάρχει στο ρομπότ την στιγμή που αυτές υπολογίζονται.

Στην παρούσα διπλωματική εργασία η χρήση των αισθητήρων υπέρηχων είναι πολύ σημαντική προκειμένου να εντοπίζονται τα εμπόδια τα οποία βρίσκονται στον προς εξέταση χώρο.

1.4 Κάμερα

Το έντροχο ρομπότ διαθέτει μια κάμερα η οποία βρίσκεται στο πάνω μέρος του ρομπότ και μπορεί να βοηθήσει σε πάρα πολλές εφαρμογές, μια εκ των οποίων είναι ότι παρέχει ολοκληρωμένη αντίληψη του χώρου είτε υπό μορφή φωτογραφίας είτε υπό μορφή εικονολήψεως (βίντεο). Τα δεδομένα των φωτογραφιών και της εικονολήψεως μπορούν να επεξεργαστούν κατάλληλα.

Η κάμερα που χρησιμοποιείται στο έντροχο ρομπότ είναι η SONY EVI D30. Διαθέτει φακό 12X ZOOM με 2 μοτέρ τα οποία της επιτρέπουν την περιστροφική κίνηση τόσο σε οριζόντιο όσο και σε κατακόρυφο άξονα. Στον κατακόρυφο άξονα έχει άνοιγμα 200^{o} και μέγιστη ταχύτητα 80^{o}/sec. Στον οριζόντιο άξονα έχει άνοιγμα 50^{o} και μέγιστη γωνιακή ταχύτητα 50^{o}/sec.

ΚΕΦΑΛΑΙΟ 2
Προσομοίωση

2.1 Εισαγωγή

Στο κεφάλαιο αυτό θα ασχοληθούμε με το πρώτο στάδιο της υλοποίησης του λογισμικού που αναπτύχθηκε στην παρούσα εργασία. Στο πρώτο στάδιο αναπτύχθηκε ένα πρόγραμμα που προσομοιώνει την κίνηση και την λήψη των δεδομένων του έντροχου ρομπότ καθώς και την επεξεργασία των δεδομένων.

Η προσομοίωση έγινε σε γλώσσα προγραμματισμού C++. Σε αυτό το επίπεδο προγραμματισμού αναπτύχθηκε ο αλγόριθμος κίνησης και ο αλγόριθμος λήψης δεδομένων από τους αισθητήρες υπερήχων. Ο αλγόριθμος προσομοίωσης είναι σημαντικός, διότι με τον τρόπο αυτό στάθηκε δυνατό να παρατηρηθεί η συμπεριφορά της κίνησης του ρομπότ σε οποιασδήποτε μορφής δωμάτιο. Επίσης όλες οι εξελίξεις οι οποίες έγιναν μετέπειτα στο κυρίως πρόγραμμα, βασίστηκαν πάνω στον αλγόριθμο προσομοίωσης.

Αρχικά υπήρξε προσπάθεια ανάπτυξης αλγόριθμου, ο οποίος δίνει τη δυνατότητα πλήρους αυτοματοποίησης της κίνησης του ρομπότ. Έτσι δεν χρειάζεται η συνεχής παρέμβαση του χειρίστη μέσω προγραμματισμού για την κίνηση του ρομπότ στο προς εξέταση δωμάτιο. Ο αλγόριθμος υπολογίζει την εκάστοτε διαδρομή σε οποιοδήποτε μορφής δωμάτιο χωρίς την παρέμβαση κάθε φορά του χειριστή.

Στο σημείο αυτό πρέπει να σημειωθεί, ότι η μορφή του αλγορίθμου επιτρέπει στο ρομπότ, την εύρεση της εξόδου όταν βρίσκεται μέσα σε κάποιον λαβύρινθο χωρίς να είναι απαραίτητη η γνώση της μορφής του λαβυρίνθου.

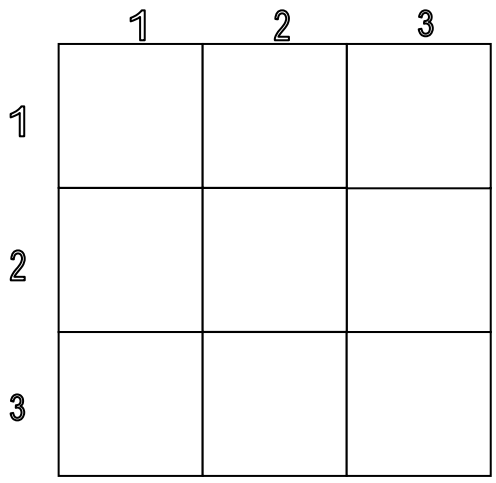
2.2 Ανάλυση αλγόριθμου

2.2.1 Αναπαράσταση χώρου

Στη παράγραφο αυτή παρουσιάζεται ο εικονικός χωρισμός του δωματίου σε υποχώρους λαμβάνοντας υπόψη.

- τις διαστάσεις του τροχοφόρου ρομπότ ATRV-mini.
- τις διαστάσεις του προς χαρτογράφηση δωματίου.

Για την ανάπτυξη του αλγόριθμου χωρίστηκε ο προς εξέταση χώρος σε υποχώρους. Η ύπαρξη των υποχώρων είναι απαραίτητη για την υλοποίηση και τη σωστή λειτουργία του αλγορίθμου κίνησης του ρομπότ. Ο προγραμματισμός του αλγορίθμου, γίνετε ευκολότερος με τη μετατροπή του χώρου ως πίνακα. Ο τρόπος που αριθμούμε τις γραμμές και τις στήλες του πίνακα παρουσιάζονται στο Σχήμα 2.1 Στο Σχήμα 2.1 παρουσιάζεται ένα παράδειγμα χωρισμού ενός δωματίου σε υποχώρους.



Σχήμα 2.1: Αναπαράσταση χωρισμένου δωματίου.

Αυτό πραγματοποιήθηκε για τους παρακάτω λόγους.

- Καλύτερη ανάπτυξη του αλγόριθμου κίνησης .
- Ταχύτερη επεξεργασία των δεδομένων από τους αισθητήρες υπερήχων.
- Ακριβέστερες μετρήσεις από τους αισθητήρες υπερήχων.

Από τις διαστάσεις του ρομπότ υπολογίστηκαν οι διαστάσεις του κάθε υποχώρου. Οι διαστάσεις προκύπτουν όπως φαίνεται παρακάτω. Οι διαστάσεις του ρομπότ είναι 60,5cm μήκος και 53,3cm πλάτος. Το ρομπότ όταν χρειαστεί να πραγματοποιήσει στροφή, τότε αυτή γίνεται γύρω από το κέντρο του. Επομένως το ρομπότ για μη συγκρουστεί με κάποιο εμπόδιο όταν στρίβει πρέπει να υπάρχει ελεύθερος χώρος σε μια ακτίνα ίση με το μισό της διαγωνίου του. Με βάση το Πυθαγόρειο θεώρημα η διαγώνιος του ρομπότ είναι 80,6cm. Επομένως οι διαστάσεις των προς εξέταση υποχωρών θα πρέπει να είναι μεγαλύτερες από την διάμετρο του κύκλου, που διαγράφει το ρομπότ σε μια πλήρη περιστροφή η οποία είναι 80,6cm. Για τον λόγο αυτό θεωρήθηκε ότι οι διαστάσεις των υποχώρων θα πρέπει να είναι 90cm x 90cm ώστε να είμαστε βέβαιοι ότι δεν υπάρχει καμία περίπτωση σύγκρουσης του τροχοφόρου ρομπότ εντός του υποχώρου.

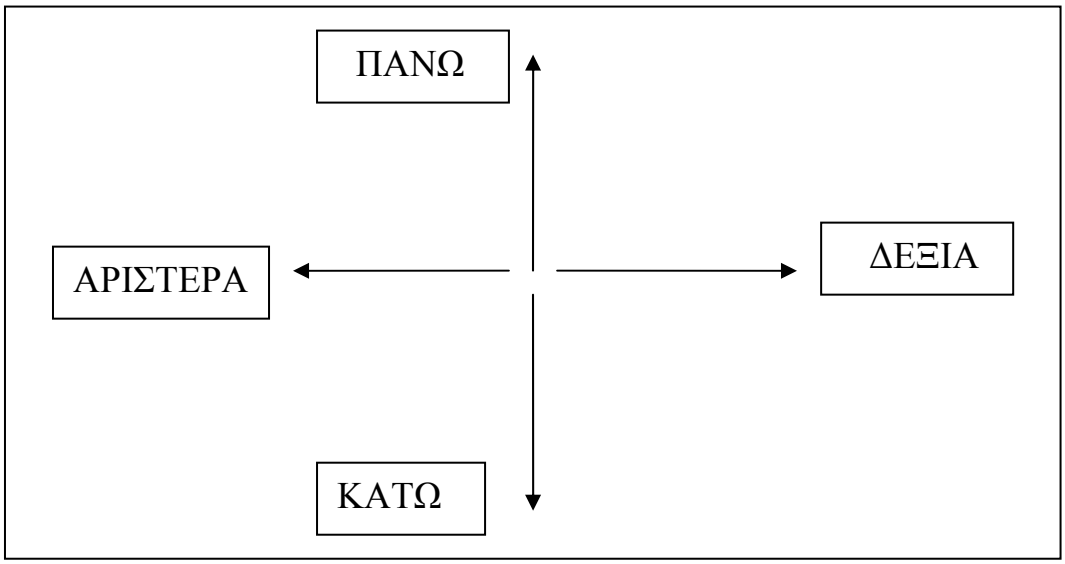
Έτσι αναπαριστάται ο προς εξέταση χώρος σε ένα τετραγωνικό πίνακα διαστάσεων K*K Η διάσταση **K** του πίνακα υπολογίζεται από το πηλίκο της διαίρεσης **N/0,9**. Με το γράμμα **N** συμβολίζεται η διάσταση εκείνη του δωματίου στο επίπεδο η οποία είναι η μεγαλύτερη από όλες και μετράται σε μέτρα. Το αριθμητικό μέγεθος **0,9** είναι η διάσταση του κάθε υποχώρου μετρούμενη σε μέτρα που εξετάζει το ρομπότ.

Αν **N- 0,9K > 0** δηλαδή ο προς εξέταση χώρος δεν διαιρείται ακριβώς σε υποχώρους διαστάσεων 0,9*0,9 τότε το νέο **K** θα είναι το ακέραιο μέρος του πηλίκου **N/0,9** προστιθέμενο κατά μια μονάδα. Αυτό έγινε ώστε να μην αποκοπεί κάποιο τμήμα του προς εξέταση χώρου επομένως **K=K+1** αλλιώς το **K** μένει ως έχει.

2.2.2 Αλγόριθμος κίνησης

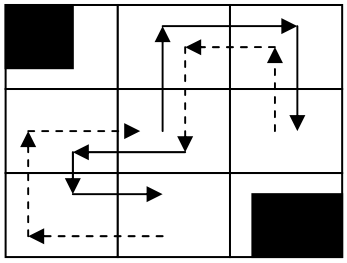
Σε κάθε θέση εξετάζεται αν μπορεί το ρομπότ να πάει προς τα πάνω. Αν "ναι" προχωράει στη νέα θέση αν "όχι" εξετάζει αν μπορεί να πάει στη αριστερή θέση. Αν "ναι" πηγαίνει στη νέα θέση δηλαδή προς τα αριστερά αν "όχι" εξετάζει αν μπορεί να κινηθεί δεξιά αν "ναι" θα πάει στη νέα θέση η θα εξετάσει αν μπορεί να πάει κάτω.

Οι κατευθύνσεις πάνω, κάτω, δεξιά και αριστερά παρουσιάζονται στο Σχήμα 2.2. Οι ορισμοί αυτοί παραμένουν οι ίδιοι πάντα όπως φαίνεται στο σχήμα, άσχετα με την θέση του ρομπότ σε σχέση με αυτές. Δηλαδή ακόμα και αν το ρομπότ κινείται προς την αριστερή κατεύθυνση η θέση «πάνω» είναι αυτή που φαίνεται στο σχήμα και θα είναι η πρώτη που θα εξεταστεί.



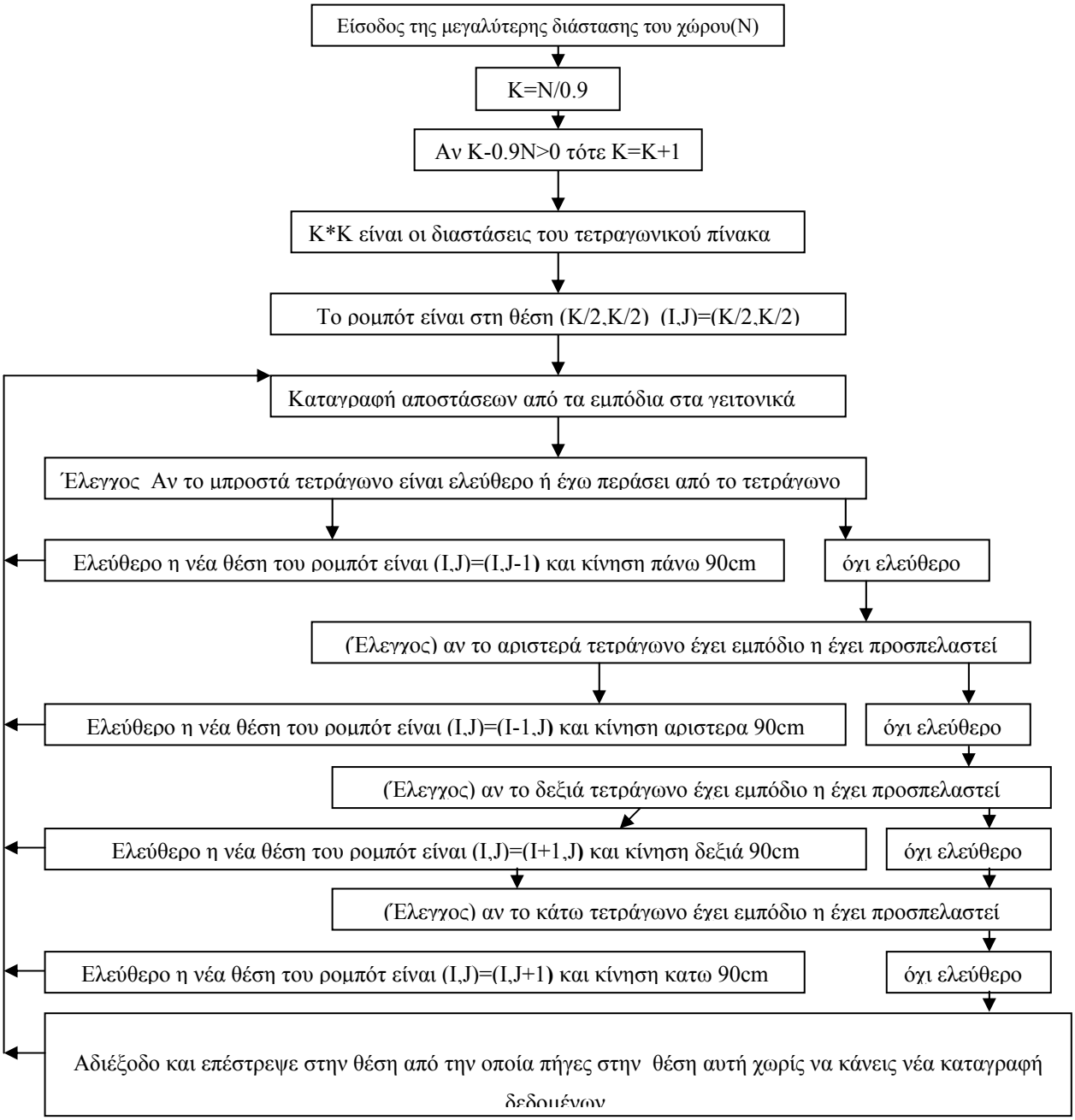
Σχήμα 2.2: Αναπαράσταση των κινήσεων του έντροχου ρομπότ ATRV-mini.

Όταν το ρομπότ βρεθει σε υποχώρο του οποίου όλοι οι γειτονικοί υποχώροι περιέχουν εμπόδιο ή το ρομπότ έχει περάσει ήδη από αυτούς, τότε ο υποχώρος οδηγεί σε αδιέξοδο. Τότε το ρομπότ επιστρέφει στον αμέσως προηγούμενο υποχώρο πριν οδηγηθεί σε αδιέξοδο. Στο νέο υποχώρο ελέγχει τις υπόλοιπες δυνατές κατευθύνσεις με την σειρά με την οποία αναφέρθηκε παραπάνω (Σχήμα 2.2). Αν υπάρχει δυνατή κατεύθυνση τότε ο αλγόριθμος συνεχίζει κανονικά. Σε αντίθετη περίπτωση, επιστρέφει στον προηγούμενο υποχώρο από τον υποχώρο που βρίσκετε εκείνη την στιγμή και ούτω καθεξής, μέχρι να βρει κάποια θέση στην οποία έχει πάει και υπάρχει κατεύθυνση η οποία δεν έχει προσπελαστεί. Όταν όλοι οι υποχώροι προσπελαστούν κάθε υποχώρος οδηγεί σε αδιέξοδο. Τότε το ρομπότ επιστρέφει σε προηγούμενους υποχώρους και φτάνοντας στην αρχική του θέση ο αλγόριθμος τερματίζεται. Ένα τέτοιο παράδειγμα παρουσιάζεται στο Σχήμα 2.3 Με συνεχή γραμμή βέλους συμβολίζεται η πορεία του ρομπότ μέχρι να οδηγηθεί σε αδιέξοδο και με διακεκομμένη η επιστροφή σε προηγούμενους υποχώρους.



Σχήμα 2.3: Αναπαράσταση πορείας του ρομπότ σε ένα δωμάτιο.

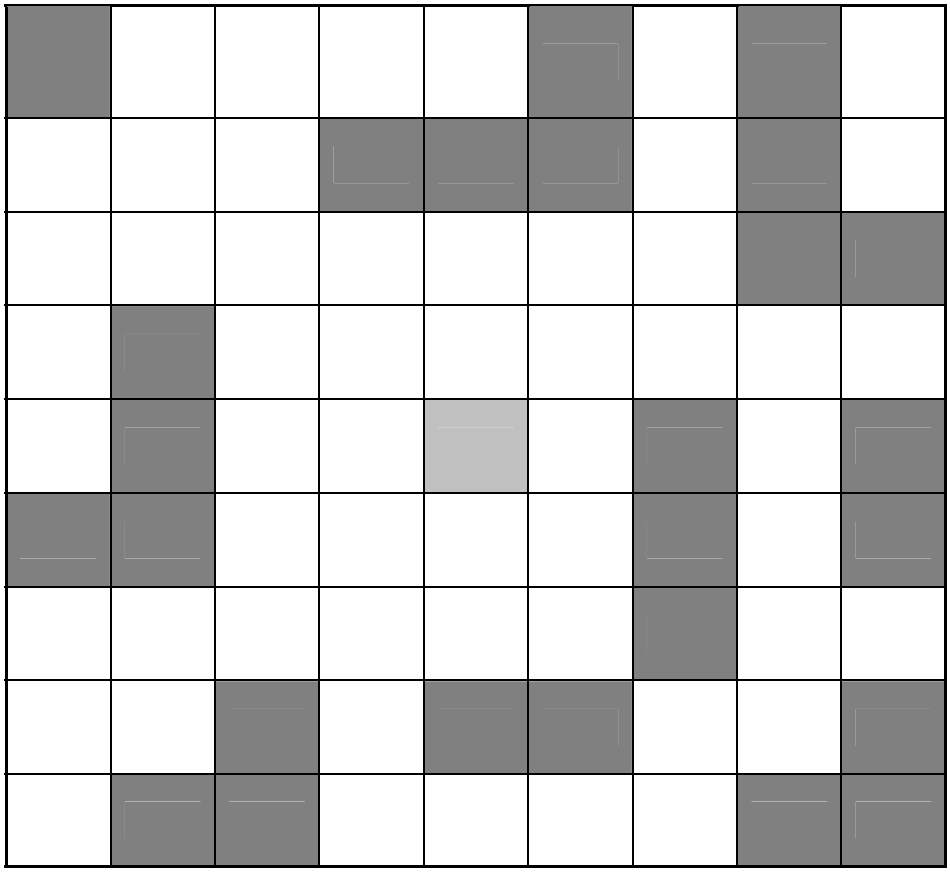
Στο Σχήμα 2.4 παρουσιάζεται το διάγραμμα ροής του παραπάνω αλγόριθμου.



Σχήμα 2.4: Διάγραμμα ροής του αλγορίθμου κίνησης .

2.3 Εφαρμογή αλγορίθμου

Πριν την εργαστηριακή εφαρμογή του αλγορίθμου κίνησης έγινε προσομοίωση του συστήματος προκειμένου να μελετηθεί η αποτελεσματικότητά του. Έστω ένα εικονικό δωμάτιο διαστάσεων 8m*8m.Όπως αναφέρθηκε στην παράγραφο 2.2.1, το δωμάτιο αναπαριστάται ως πίνακας. Οι διαστάσεις του πίνακα υπολογίζονται από το πηλίκο της διαίρεσης N/0,9 όπου N=8m και είναι ίσο με 8/0,9=8,888.Επειδη το αποτέλεσμα της διαίρεσης είναι δεκαδικός προκύπτει ότι οι διαστάσεις του πίνακα είναι 9*9. Στο Σχήμα 2.5 παρουσιάζεται το χωρισμένο εικονικό δωμάτιο σε υποχώρους, μαζί με τα χρωματισμένα με γκρίζο χρώμα εμπόδια



Σχήμα 2.5: Αναπαράσταση του εικονικού δωματίου στο οποίο έγινε η εφαρμογή του αλγόριθμου κίνησης

Η αναλυτική παρουσίαση των βημάτων του ρομπότ κατά την σάρωση του εικονικού δωματίου απαντάται στα τελικά αποτελέσματα τα οποία παρουσιάζονται στο Σχήμα 2.6. Σε όλους τους προσβάσιμους υποχώρους αναγράφεται μια σειρά αριθμών

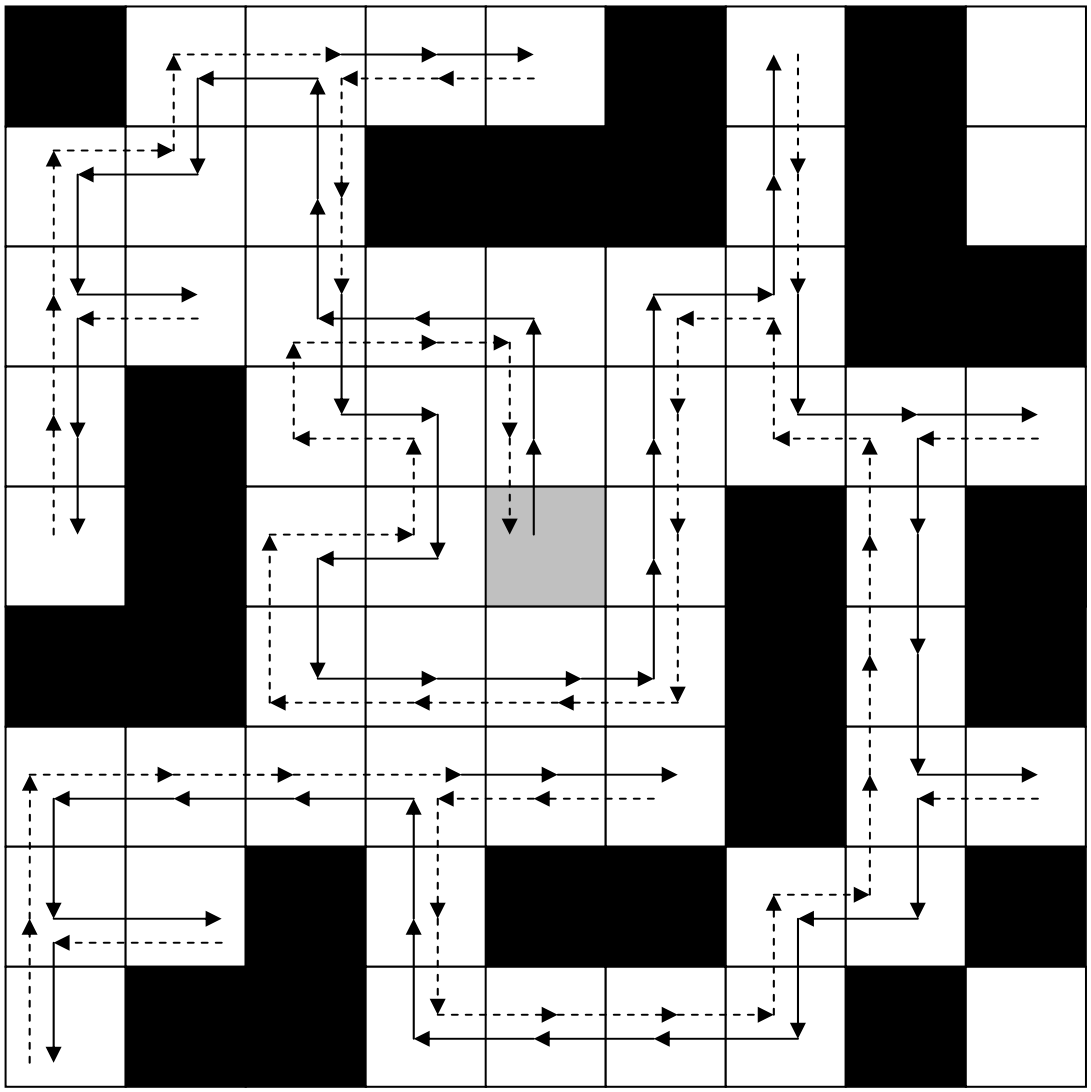
(X,Y,Z,….). Το πλήθος των αριθμών δηλώνουν τον αριθμό των διελεύσεων και η αύξουσα σειρά των αριθμών δηλώνει την πορεία με του ρομπότ με βήματα στο χώρο.

	7,19	6,20,24	21,23	22		40		
9,17	8,18	5,25				39,41		
10,12,16	11	4,26,100	3,101	2,102	37,89	38,42,88		
13,15		27,99	28,98	1,103	36,90	43,87	44,46,86	45
14		30,96	29,97	0,104	35,91		47,85	
		31,95	32,94	33,93	34,92		48,84	
62,68	61,69	60,70	59,71,75	72,74	73		49,51,83	50
63,65,67	64		58,76			53,81	52,82	
66			57,77	56,78	55,79	54,80		

Σχήμα 2.6: Αρίθμηση των βημάτων που έκανε το ρομπότ κατά την εφαρμογή του αλγορίθμου στο εικονικό δωμάτιο.

Αναλυτικότερα η πορεία του ρομπότ έγινε ως εξής. Το ρομπότ αρχικά τοποθετείται στο κέντρο (5,5) του δωματίου και εξετάζει αν στον πάνω γειτονικό του υποχώρο υπάρχει εμπόδιο. Στο υποχώρο αυτό δεν υπάρχει εμπόδιο όπως φαίνεται στο σχήμα έτσι παει πάνω και κάνει το βήμα ένα (1). Τώρα το ρομπότ είναι στο υποχώρο (4,5) και εξετάζει αν μπορεί να παει στο πάνω γειτονικό υποχώρο. Ο πάνω γειτονικά υποχώρος δεν έχει εμπόδιο άρα μπορεί να μετακινηθεί στο υποχώρο αυτό και κάνει το βήμα δυο(2). Τώρα το ρομπότ είναι στο υποχώρο (3,5) και εξετάζει αν μπορεί να παει στο πάνω γειτονικό υποχώρο. Ο πάνω γειτονικά υποχώρος έχει εμπόδιο άρα δεν μπορεί να μετακινηθεί στο υποχώρο αυτό. Η επόμενη επιλογή που έχει είναι ο

αριστερός γειτονικός υποχώρος ο οποίος δεν περιέχει εμπόδιο. Το ρομπότ επομένως μετακινείται στον αριστερό υποχώρο και πραγματοποιείται το βήμα τρία(3). Τώρα το ρομπότ είναι στον υποχώρο(3,4) και εξετάζει αν μπορεί να παει στο πάνω γειτονικό υποχώρο. Ο πάνω γειτονικά υποχώρος έχει εμπόδιο άρα δεν μπορεί να μετακινηθεί στον υποχώρο αυτό. Η επόμενη επιλογή που έχει είναι ο αριστερός γειτονικός υποχώρος ο οποίος δεν περιέχει εμποδιο. Το ρομπότ επομένως μετακινείται στον αριστερό υποχώρο και πραγματοποιείται το βήμα τέσσερα(4). Τώρα το ρομπότ είναι στον υποχώρο(3,3) και εξετάζει αν μπορεί να παει στο πάνω γειτονικό υποχώρο. Ο πάνω γειτονικά υποχώρος δεν έχει εμπόδιο άρα μπορεί να μετακινηθεί στο υποχώρο αυτό και πραγματοποιείται το βήμα πέντε(5). Έτσι συνεχίζει την πορεία του το ρομπότ. Στο δέκατο τέταρτο βήμα το ρομπότ έχει οδηγηθεί σε αδιέξοδο και όπως φαίνεται από το Σχήμα 2.6 το ρομπότ επιστρέφει στη θέση του βήματος δεκατρία(13). Από την θέση αυτή για τον ίδιο λόγο το ρομπότ πηγαίνει στην αμέσως προηγούμενη θέση. Η αναπαράσταση της πορείας του ρομπότ παρουσιάζεται στο Σχήμα 2.7.



Σχήμα 2.7: Γραφική αναπαράσταση της πορείας που έκανε το ρομπότ στο εικονικό δωμάτιο.

2.4 Αλγόριθμος αισθητήρων υπερήχων

Στην παράγραφο αυτή παρουσιάζεται ο αλγόριθμος των αισθητήρων υπερήχων. Οι αισθητήρες λειτουργούν σε όλη την διάρκεια που πραγματοποιείται η συλλογή των δεδομένων για την χαρτογράφηση. Αυτό δημιουργεί τα παρακάτω προβλήματα.

- Η απόσταση των εμποδίων από το ρομπότ μεταβάλλεται όσο το όχημα κινείται. Έτσι δεν υπάρχει ακριβής εγγραφή της απόσταση του ρομπότ από σταθερό εμπόδιο.

- Εξ αιτίας της γεωμετρίας των αισθητήρων υπερήχου υπάρχει πιθανότητα οι θέσεις των εμποδίων του χώρου να υπολογίζονται σε διαφορετικές θέσεις από εκείνες που πραγματικά ευρίσκονται ανάλογα με την σχετική θέση του ρομπότ ως προς το εμπόδιο.

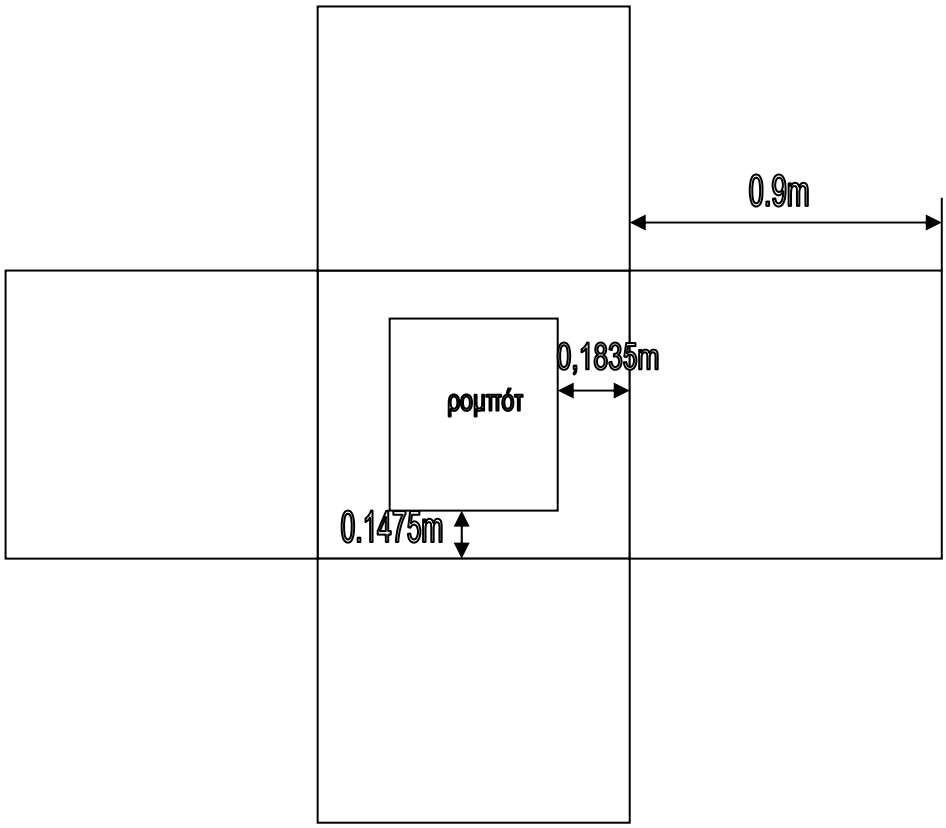
Οι παραπάνω δυσκολίες αντιμετωπίστηκαν με τον παρακάτω τρόπο.

Πρώτον, το όχημα θα κάνει μέτρηση και εγγραφή της μέτρησης, σε ακινησία. Δηλαδή σε κάθε βήμα θα σταματάει και εν συνεχεία θα μετράει την απόσταση από τα πιθανά εμπόδια. Με τον τρόπο αυτό κατορθώθηκε ο μη συνεχής υπολογισμός της απόστασης ενός αντικειμένου όσο κινείται το ρομπότ.

Δεύτερον, δεν είναι σκόπιμο να λειτουργούν όλοι οι αισθητήρες, παρά μόνο αυτοί που είναι χρήσιμοι και είναι απαραίτητοι. Δηλαδή προκείμενου να επιτευχθεί ακριβέστερη μέτρηση της απόσταση θα λειτουργούν από κάθε πλευρά του οχήματος από ένα ζεύγος αισθητήρων.

Η μέτρηση κάποιας απόστασης από το εμπρός μέρος του ρομπότ, πραγματοποιείται από τους αισθητήρες 0 και 23. Η απόσταση από το κάτω μέρος του ρομπότ, υπολογίζεται χρησιμοποιώντας τους αισθητήρες 11 και 12. Ο υπολογισμός της απόστασης κάποιου εμποδίου από δεξιά, υπολογίζεται χρησιμοποιώντας τους αισθητήρες 18 και 19. Και αντίστοιχα από τα αριστερά, οι αισθητήρες 5 και 6.

Τέλος η απόσταση υπολογίζεται με την μέση τιμή των τιμών από το ζεύγος αισθητήρων υπερήχου. Επίσης κάθε φορά που τα ζεύγη αισθητήρων (4,5) και (18,17) υπολογίζουν κάποια απόσταση πέρα από το 1.0835m, τότε δεν υπάρχει εμπόδιο από την εκάστοτε πλευρά που βρίσκονται τα ζεύγη αισθητήρων. Αντίστοιχα τα ζεύγη των αισθητήρων (0,23) και (11,12) αν μετρήσουν απόσταση μεγαλύτερη από 1,0475m. Σε κάθε άλλη περίπτωση υπάρχει εμπόδιο. Στο Σχήμα 2.8 παρουσιάζονται οι παραπάνω αποστάσεις.



Σχήμα 2.8:Αναπαράσταση των μέγιστων αποστάσεων.

2.5 Μορφή αποτελεσμάτων

Τα αποτελέσματα έχουν την παρακάτω μορφή στον πίνακα της προσομοίωσης. Το πρώτο νούμερο είναι το μέγεθος του πίνακα το οποίο δείχνει και το πλήθος των υποχώρων. Αν για παράδειγμα ο πρώτος αριθμός είναι 4 τότε το πλήθος των υποχώρων είναι 4*4=16. Κάθε υποχώρος έχει αρχικά την τιμή 0 η 1. Είναι 0 εάν είναι κενός χώρος και 1 αν υπάρχει εμπόδιο. Κατόπιν υπάρχουν τέσσερεις τιμές οι οποίες δηλώνουν την απόσταση του εμποδίου από το ρομπότ, μετράται σε μέτρα και αντιστοιχούν σε κάθε πλευρά του υποχώρου. Αν υπάρχει κενός υποχώρος τότε μπαίνουν 4 ενδεικτικές τιμές (2,00000). Σε περίπτωση ύπαρξης εμποδίου τότε καταχωρούνται οι ακριβείς αποστάσεις εντός του υποχώρου. Για παράδειγμα από το αποτέλεσμα.

- 0 2,00000 2,00000 2,00000 2,00000

συμπεραίνεται ότι δεν υπάρχει εμπόδιο στο συγκεκριμένο υποχώρο. Ενώ από το παρακάτω.

- 1 0,55000 0,60000 2,00000 0,35200

συμπεραίνεται ότι στον εξεταζόμενο υποχώρο υπάρχει εμπόδιο και αυτό το εμπόδιο βρίσκεται σε απόσταση 55cm από την πάνω πλευρά του υποχώρου. Εν συνεχεία 60cm από την αριστερή πλευρά, απέχει 0 cm από την δεξιά πλευρά και 35,2 cm από την κάτω πλευρά. Επίσης οι υποχώροι αριθμούνται από τα αριστερά προς τα δεξιά και από πάνω προς τα κάτω.

2.6 Εφαρμογή τελικού αλγόριθμου

Στο πρόγραμμα της προσομοίωσης συμπληρώθηκαν από τον χρήστη οι αποστάσεις των εμποδίων εντός των υποχώρων για να παρατηρηθεί η αποτελεσματικότητα της παρουσίασης των δεδομένων.

9

δείκτης εμποδίου	απόσταση από την πάνω πλευρά του υποχώρου	απόσταση από την αριστερή πλευρά του υποχώρου	απόσταση από την δεξιά πλευρά του υποχώρου	απόσταση από την κάτω πλευρά του υποχώρου
1	2.0000	2.0000	0.3165	0.2525
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
1	2.0000	0.3665	0.3165	2.0000
0	2.0000	2.0000	2.0000	2.0000
1	2.0000	0.3665	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000

Πίνακας 2.1: Αποτελέσματα του προγράμματος προσομοίωσης.

δείκτης εμποδίου	απόσταση από την πάνω πλευρά του υποχώρου	απόσταση από την αριστερή πλευρά του υποχώρου	απόσταση από την δεξιά πλευρά του υποχώρου	απόσταση από την κάτω πλευρά του υποχώρου
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
1	0.3025	0.3665	2.0000	0.2525
1	0.3025	2.0000	2.0000	0.2525
1	2.0000	2.0000	0.3165	0.2525
0	2.0000	2.0000	2.0000	2.0000
1	2.0000	0.3665	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
1	2.0000	0.3665	2.0000	0.2525
1	2.0000	2.0000	0.3165	0.2525
0	2.0000	2.0000	2.0000	2.0000
1	0.3025	0.3665	0.3165	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000

Πίνακας 2.1: Παρουσίαση αποτελεσμάτων προσομοίωσης (συνέχεια).

δείκτης εμποδίου	απόσταση από την πάνω πλευρά του υποχώρου	απόσταση από την αριστερή πλευρά του υποχώρου	απόσταση από την δεξιά πλευρά του υποχώρου	απόσταση από την κάτω πλευρά του υποχώρου
1	2.0000	0.3665	0.3165	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
1	0.3025	0.3665	0.3165	2.0000
0	2.0000	2.0000	2.0000	2.0000
1	0.3025	0.3665	2.0000	2.0000
1	0.3025	2.0000	2.0000	0.2525
1	2.0000	2.0000	0.3165	0.2525
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
1	2.0000	0.3665	0.3165	2.0000
0	2.0000	2.0000	2.0000	2.0000
1	2.0000	0.3665	0.3165	0.2525
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
1	2.0000	0.3665	0.3165	0.2525
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000

Πίνακας 2.1: Παρουσίαση αποτελεσμάτων προσομοίωσης (συνέχεια).

δείκτης εμποδίου	απόσταση από την πάνω πλευρά του υποχώρου	απόσταση από την αριστερή πλευρά του υποχώρου	απόσταση από την δεξιά πλευρά του υποχώρου	απόσταση από την κάτω πλευρά του υποχώρου
0	2.0000	2.0000	2.0000	2.0000
1	0.3025	0.3665	0.3165	2.0000
0	2.0000	2.0000	2.0000	2.0000
1	0.3025	0.3665	2.0000	0.2525
1	0.3025	2.0000	0.3165	0.2525
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
1	0.3025	0.3665	0.3165	2.0000
0	2.0000	2.0000	2.0000	2.0000
1	0.3025	0.3665	2.0000	2.0000
1	2.0000	2.0000	0.3165	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
1	0.3025	0.3665	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000

Πίνακας 2.1: Παρουσίαση αποτελεσμάτων προσομοίωσης (συνέχεια).

ΚΕΦΑΛΑΙΟ 3
Πειραματική εφαρμογή

3.1 Εισαγωγή

Στο κεφάλαιο αυτό θα γίνει παρουσίαση της εφαρμογής του αλγόριθμου που αναλύθηκε στο προηγούμενο κεφαλαίο σε πραγματικές συνθήκες. Η εφαρμογή αυτή έγινε στο εργαστήριο για τρεις χώρους. Ο πρώτος χώρος είναι κενός ενώ στους άλλους δυο τοποθετήθηκαν εμπόδια. Στο εργαστήριο υλοποιήθηκαν άλλα δυο λογισμικά (Navigdata, Map viewer). Το πρώτο καταγράφει την πορεία της κίνησης του ρομπότ στο χώρο. Το δεύτερο υλοποιεί την γραφική αναπαράσταση του χώρου. Αυτό έγινε διότι η μορφή των αποτελεσμάτων όπως παρουσιάστηκε στο κεφάλαιο 2 δεν αποτελούν γραφική αναπαράσταση.

Η πολυπλοκότητα του αλγορίθμου κίνησης του ρομπότ προκαλεί σφάλματα στη προγραμματισμένη κίνηση του ρομπότ. Όταν χαρτογραφείται μεγάλος χώρος τότε το ρομπότ χρειάζεται να πραγματοποίηση μεγάλο αριθμό στροφών. Εξαιτίας κακής ποιότητας εδάφους ή ελλιπής συντήρησης ελαστικών, οι στροφές προς τα δεξιά η προς τα αριστερά που διαγράφει το ρομπότ δεν είναι ακριβής. Αυτό έχει ως αποτέλεσμα το ρομπότ να βγει εκτός της προγραμματισμένης πορείας του. Αν για παράδειγμα το ρομπότ στρίβει με απόκλιση 5^o, τότε μετά από 10 στροφές προς την ίδια φορά θα έχει απόκλιση 40^o από την πορεία που θα έπρεπε να έχει ακολουθήσει.

3.2 Λογισμικό καταγραφής της πορείας του ρομπότ

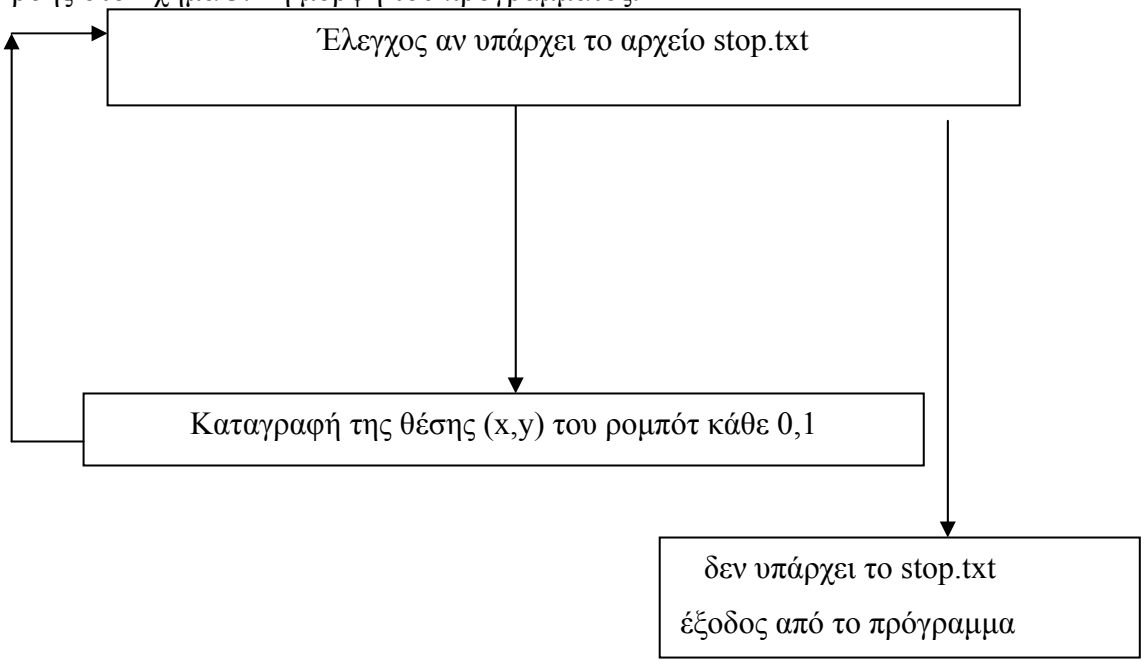
Το λογισμικό καταγραφής της πορείας του ρομπότ λειτουργεί παράλληλα με την κίνηση του ρομπότ. Η καταγραφή της πορείας έγινε με διαφορετικό πρόγραμμα, διότι η δομή του προγράμματος συλλογής πληροφοριών δεν επέτρεπε την ενσωμάτωση του σε αυτό. Το πρόβλημα ήταν, ότι κατά την διάρκεια της κίνησης το κυρίως πρόγραμμα στο έντροχο όχημα χρησιμοποιεί την εντολή sleep. Αυτό σημαίνει, ότι όσο κινείται το ρομπότ δεν πραγματοποιείται καμία άλλη εντολή. Δηλαδή δεν θα

ήταν δυνατόν να εκτελεστούν οι εντολές εκείνες οι οποίες καταγράφουν την πορεία του ρομπότ.

Τα δύο προγράμματα λειτουργούν ως εξής. Πρώτα γίνεται εφαρμογή του προγράμματος κίνησης και καταγραφής δεδομένων μέχρι του σημείου όπου εισάγονται οι διαστάσεις του προς εξέταση χώρου. Μέχρι το σημείο εκείνο το ρομπότ είναι σε ακινησία. Κατόπιν γίνεται εφαρμογή του προγράμματος καταγραφής της πορείας του ρομπότ και εν συνεχεία πατάμε enter για την εκτέλεση του προγράμματος κίνησης και καταγραφής δεδομένων. Η παραπάνω διαδικασία έγινε καθαρά για προγραμματιστικούς λόγους.

Με δοκιμές υπολογίστηκαν οι τιμές της ταχύτητας, οι οποίες παρουσιάζουν μικρή απόκλιση. Παρόλα αυτά όταν το ρομπότ κινείται σε διαφορετικές επιφάνειες παρουσιάζονται προβλήματα κατά την συνεχόμενη αλλαγή πορείας του ρομπότ.

Πρέπει να σημειωθεί ότι, όσο περισσότερο χρόνο κινείται το ρομπότ τόσο πιο μεγάλη είναι η πιθανότητα το οδόμετρο να καταγράφει λάθος θέση λόγο ολισθηρότητας του δαπέδου η ολισθηρότητας των ελαστικών. Παρακάτω παρουσιάζεται σε διάγραμμα ροής στο Σχήμα 3.1 η μορφή του προγράμματος.



Σχήμα 3.1: Διάγραμμα ροής προγράμματος καταγραφής της πορείας.

Το αρχείο stop.txt υπάρχει για τον εξής λόγο. Όταν αρχίζει να λειτουργεί το κυρίως πρόγραμμα, τότε μέσο του κώδικα δημιουργεί το αρχείο stop.txt έτσι ώστε το δευτερεύον πρόγραμμα καταγραφής θέσης, να γνωρίζει ότι μπορεί να εκτελείται. Σε περίπτωση που δεν υπάρχει το αρχείο αυτό, τότε το πρόγραμμα σταματά. Με τον τρόπο αυτό το δευτερεύον πρόγραμμα γνωρίζει πότε να καταγράφει την θέση του ρομπότ και πότε να σταματά

3.3 Ανάλυση αλγορίθμου χαρτογράφησης

Ο αλγόριθμος της χαρτογράφησης βασίζεται στα δεδομένα που αποθήκευσε το ρομπότ κατά την κίνηση του μέσα στο χώρο. Τα αποτελέσματα όπως αυτά αναλύονται στη παράγραφο 2.5 βοηθούν στην χαρτογράφηση. Η γραφική αναπαράσταση γίνεται ως εξής. Αρχικά ο αλγόριθμος αποθηκεύει τις διαστάσεις K*K του πίνακα που αντιστοιχούν σε κάθε χώρο. Στη συνέχεια κατασκευάζει ένα λευκό φόντο το οποίο διαιρείται σε K^2 υποχώρους. Ο αλγόριθμος αρχίζει την χαρτογράφηση από το στοιχείο (1,1) και συνεχίζει με βάση την ροή των αποτελεσμάτων, όπως αναλύθηκε στη παράγραφο 2.5. Όταν ο πρώτος αριθμός του ζεύγους αποτελεσμάτων για κάθε υποχώρο, που χαρακτηρίζει την ύπαρξη η όχι εμποδίου στον εκάστοτε υποχώρο, είναι το 0, τότε δεν έχουμε εμπόδιο και άρα ο υποχώρος παραμένει σε άσπρο φόντο. Σε αντίθετη περίπτωση η μορφή των αποτελεσμάτων του υποχώρου υποδηλώνουν αποστάσεις του αντικειμένου από το πάνω, αριστερό, δεξιό και κάτω άκρο του υποχώρου. Επομένως ο αλγόριθμος μπορεί να γνωρίζει τόσο την ύπαρξη εμποδίου όσο και να υπολογίζει τις διαστάσεις του εμποδίου.

Η οριζόντια διάσταση Λο του εμποδίου υπολογίζεται από τον τύπο

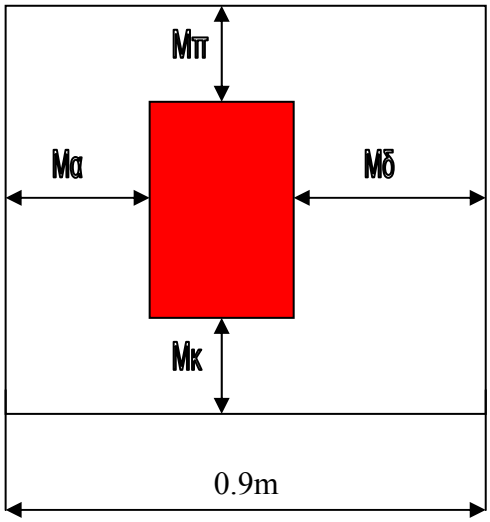
$$Λο=0,9-Μα -M_{δ}$$

Όπου Μα είναι η απόσταση του εμποδίου από το αριστερό άκρο του υποχώρου και $M_{δ}$ η απόσταση του εμποδίου από το δεξιό άκρο του υποχώρου.

Η κάθετη διάσταση Λκ του εμποδίου υπολογίζεται από τον τύπο

Λκ=0,9-Μπ –Μκ

Όπου Μπ είναι η απόσταση του εμποδίου από το πάνω άκρο του υποχώρου και Μκ η απόσταση του εμποδίου από το κάτω άκρο του υποχώρου. Στο Σχήμα 3.2 παρουσιάζονται οι διαστάσεις ενός εμποδίου μέσα σε έναν υποχώρο.



Σχήμα 3.2: Παρουσίαση των μεταβλητών ενός εμποδίου μέσα σε έναν υποχώρο.

Έτσι ο αλγόριθμος σχεδιάζει ορθογώνιο παραλληλόγραμμο διαστάσεων (Λο,Λκ) από το σημείο (Μα,Μκ) με κατεύθυνση από κάτω προς τα πάνω και από αριστερά προς τα δεξιά. Τα εμπόδια που υπάρχουν μέσα στο χώρο χρωματίζονται κόκκινα. Στην περίπτωση που κάποιος υποχώρος έχει εμπόδιο, και σε μια η περισσότερες θέσεις καταγραφής αποστάσεων υπάρχει η αρχική τιμή 2.00000, τότε αυτομάτως θεωρείται ότι το αντίστοιχο Μα, Μδ, Μπ, Μκ είναι ίσο με την τιμή 0. Η αρχική τιμή 2.0000 παραμένει αμετάβλητη όταν το ρομπότ δεν μπορεί να καταγράψει την συγκεκριμένη απόσταση. Ένα παράδειγμα είναι η ύπαρξη αντικειμένου το οποίο βρίσκεται κοντά σε τοίχο.

3.4 Εφαρμογή στο εργαστήριο

Στο εργαστήριο πραγματοποιήθηκαν 3 πειράματα.

1. Σε κενό χώρο διαστάσεων 2,7*2,7 m^2
2. Σε χώρο διαστάσεων 3,6*3,6 m^2 με αντικείμενα
3. Σε χώρο διαστάσεων 4,5*4,5 m^2 με αντικείμενα

4. Σε χώρο διαστάσεων 2,7*2,7 m^2 με αντικείμενα

Σε κάθε χώρο τοποθετήθηκαν κούτες για εμπόδια. Ο κενός χώρος χρησιμοποιήθηκε για τον έλεγχο της αποτελεσματικότητας του αλγορίθμου σε πραγματικές συνθήκες. Στους υπόλοιπους χώρους τα αποτελέσματα επεξεργάστηκαν από το πρόγραμμα γραφικής αναπαράστασης.

3.5 Χώρος διαστάσεων 3,6*3,6 m^2 με εμπόδια

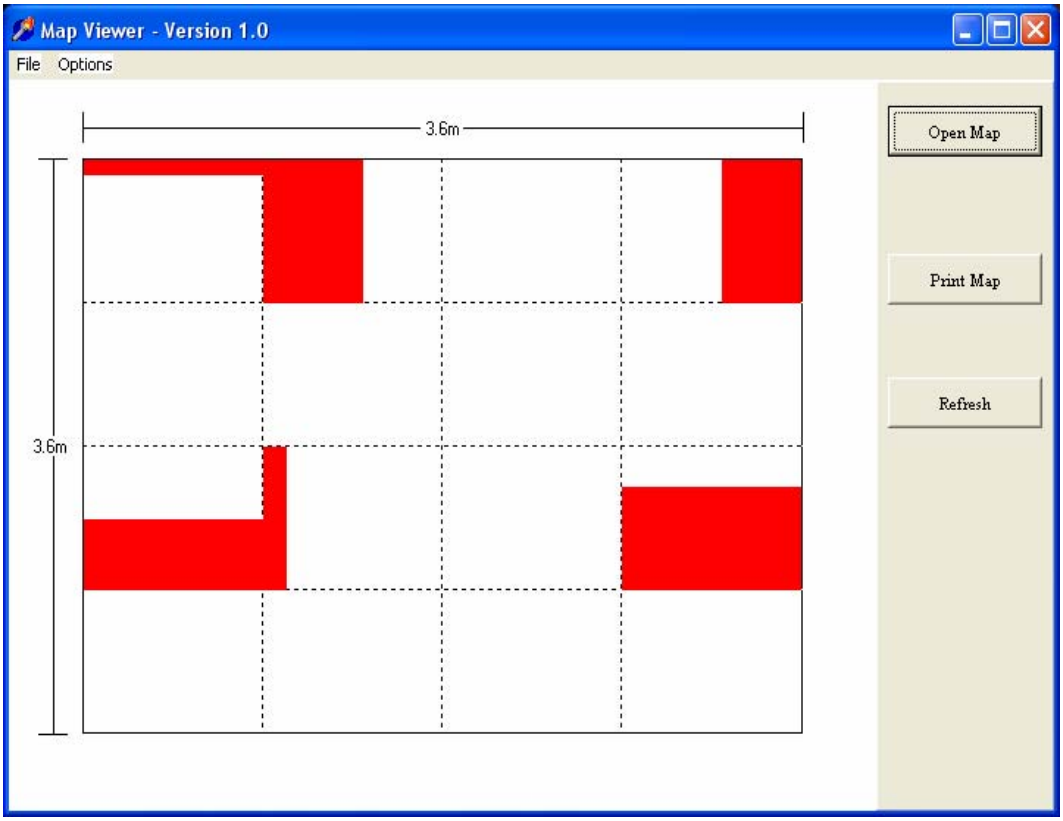
Τα αποτελέσματα που καταγράφηκαν μετά την πορεία που έκανε το ρομπότ παρουσιάζονται στο Πίνακα 3.1. Τα αποτελέσματα αυτά επεξεργάσθηκαν από το πρόγραμμα χαρτογράφησης στο οποίο γίνεται η τελική παρουσίαση του χώρου και φαίνεται στο Σχήμα 3.4. Στο Σχήμα 3.5 παρουσιάζεται σχηματικά ο πραγματικός χώρος.

4

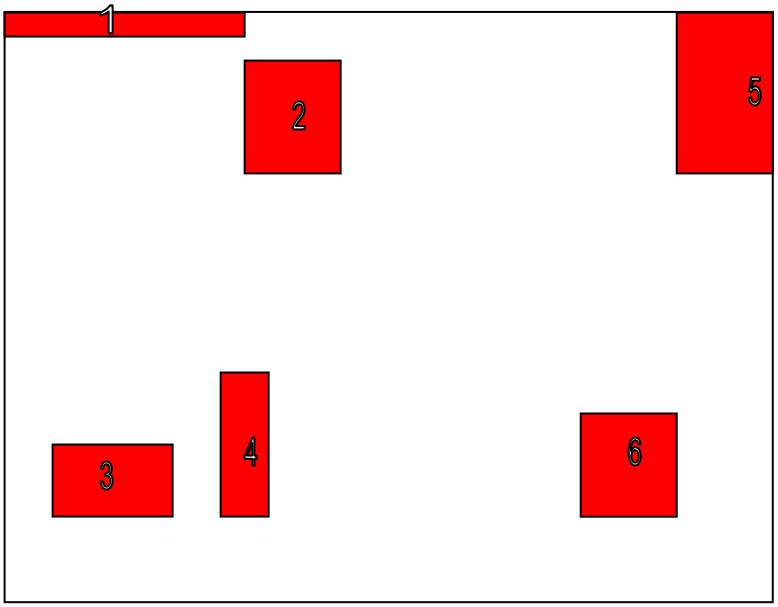
δείκτης εμποδίου	απόσταση από την πάνω πλευρά του υποχώρου	απόσταση από την αριστερή πλευρά του υποχώρου	απόσταση από την δεξιά πλευρά του υποχώρου	απόσταση από την κάτω πλευρά του υποχώρου
1	2.00000	2.00000	2.0000	0.8062
1	2.00000	2.00000	0.3971	2.0000
0	2.00000	2.00000	2.0000	2.0000
1	2.00000	0.50308	0.6821	2.0000
0	2.00000	2.00000	2.0000	2.0000
0	2.00000	2.00000	2.0000	2.0000
0	2.00000	2.00000	2.0000	2.0000
0	2.00000	2.00000	2.0000	2.0000
1	0.45235	2.00000	2.0000	0.9333
1	2.00000	2.00000	0.7802	2.0000
0	2.00000	2.00000	2.0000	2.0000
1	0.25325	2.00000	2.0000	0.7851
0	2.00000	2.00000	2.0000	2.0000

0	2.00000	2.00000	2.0000	2.0000
0	2.00000	2.00000	2.0000	2.0000
0	2.00000	2.00000	2.0000	2.0000

Πίνακας 3.1: Παρουσίαση των αποτελεσμάτων χώρου με αντικείμενα, διαστάσεων 3,6*3,6 m^2.



Σχήμα 3.4: Παρουσίαση της μορφής που έχει ο προς εξέταση κλειστός χώρος.



Σχήμα 3.5: Παρουσίαση του πραγματικού χώρου.

Συγκρίνοντας τα εμπόδια στο πραγματικό και στο χαρτογραφημένο χώρο παρατηρείται ότι τα εμπόδια στο χαρτογραφημένο χώρο διαφέρουν με αυτά που βρίσκονται στο πραγματικό χώρο. Το ρομπότ δεν καταφέρνει να μετρήσει αποστάσεις μεταξύ εμποδίων τα οποία βρίσκονται σε διπλανούς υποχώρους. Αυτό οφείλεται στο γεγονός ότι δεν είναι προσβάσιμοι από το ρομπότ οι υποχώροι από τους οποίους μπορεί να γίνει μέτρηση των αποστάσεων. Σε αυτή την περίπτωση το ρομπότ θεωρεί ότι τα εμπόδια καλύπτουν το μέρος του υποχώρου που δεν μπορεί να μετρηθεί από το ρομπότ. Αναλυτικότερα το ρομπότ βρήκε τις πραγματικές διαστάσεις του εμποδίου1. Το εμπόδιο 2 χαρτογραφήθηκε μεγαλύτερο από το κανονικό και ότι εφάπτεται με το εμπόδιο 1 το οποίο δεν συμβαίνει στην πραγματικότητα. Για το τρίτο εμπόδιο το ρομπότ δεν κατάφερε να μετρήσει την απόσταση από τον αριστερό τοίχο και δεξιό εμπόδιο. Για τον ίδιο λόγο δεν υπολογίστηκαν σωστά οι διαστάσεις του εμποδίου 6. Επειδή ο υποχώρος που βρίσκεται το εμπόδιο τρία δεν είναι προσβάσιμος ο αλγόριθμος δεν υπολογίζει τις πλήρεις διαστασεις του εμποδίου 4. Το εμπόδιο 4 στην πραγματικότητα καταλαμβάνει μέρος και από τους 2 υποχωρους. Τέλος το εμπόδιο χαρτογραφήθηκε όπως είναι στην πραγματικότητα.

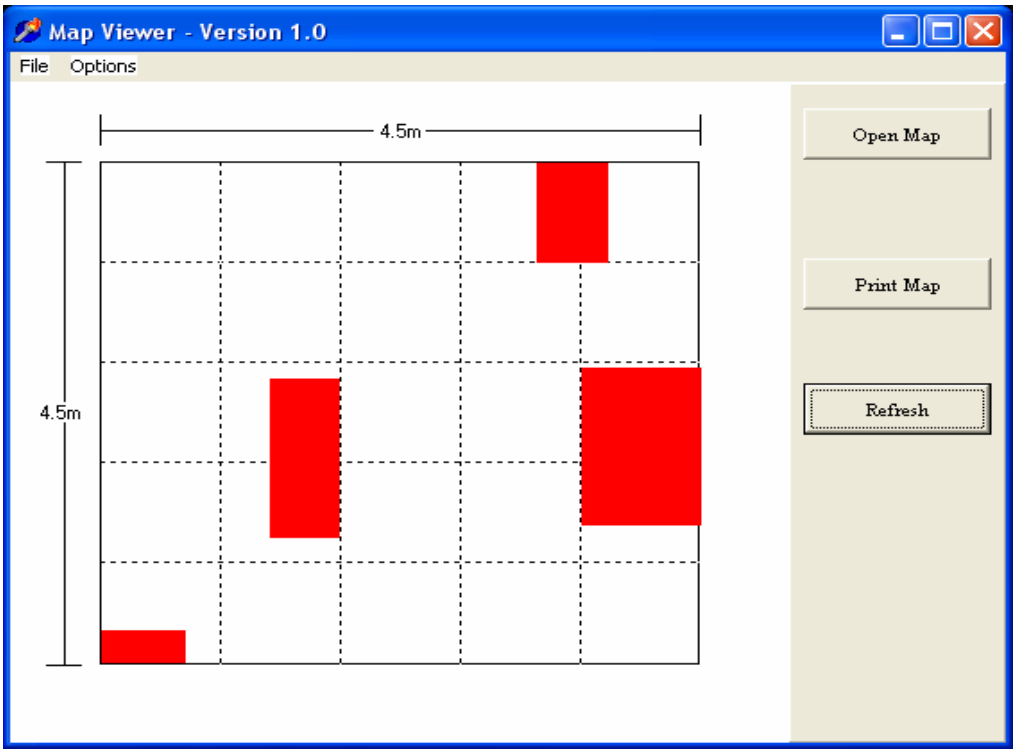
3.6 Χώρος με αντικείμενα διαστάσεων 4,5*4,5 m^2

Τα αποτελέσματα που καταγράφηκαν μετά την πορεία που έκανε το ρομπότ παρουσιάζονται στο Πίνακα 3.2. Τα αποτελέσματα αυτά επεξεργάσθηκαν από το πρόγραμμα χαρτογράφησης στο οποίο γίνεται η τελική παρουσίαση του χώρου και φαίνεται στο Σχήμα 3.6. Τέλος στο Σχήμα 3.7 παρουσιάζεται σχηματικά ο πραγματικός χώρος.

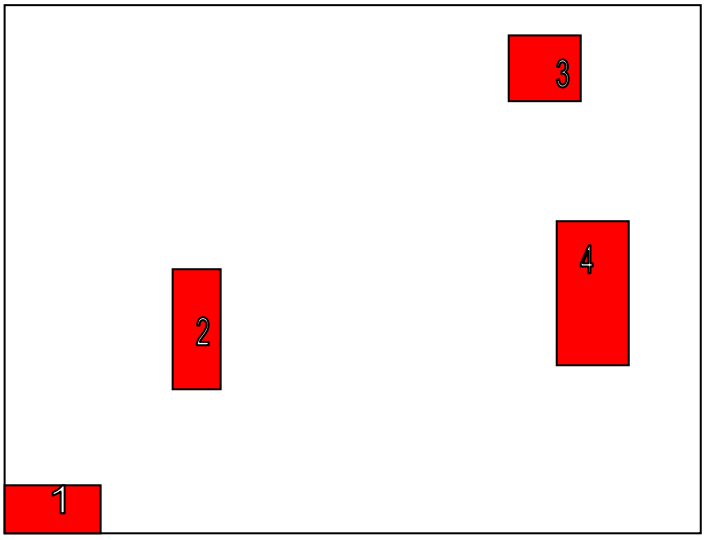
5

δείκτης εμποδίου	απόσταση από την πάνω πλευρά του υποχώρου	απόσταση από την αριστερή πλευρά του υποχώρου	απόσταση από την δεξιά πλευρά του υποχώρου	απόσταση από την κάτω πλευρά του υποχώρου
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
1	2.0000	0.5591	2.0000	2.0000
1	2.0000	2.0000	0.6931	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
1	0.1365	0.3659	0.0129	2,0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
1	0.0390	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
1	2.0000	0.3659	0.0129	0.2287
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
1	2.0000	2.0000	2.0000	0.3376
1	0.6106	2.0000	0.2650	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000
0	2.0000	2.0000	2.0000	2.0000

Πίνακας 3.2: Παρουσίαση των αποτελεσμάτων χώρου με αντικείμενα, διαστάσεων 4,5*4,5 m^2.



Σχήμα 3.6: Παρουσίαση της μορφής που έχει ο προς εξέταση κλειστός χώρος.



Σχήμα 3.7: Παρουσίαση του πραγματικού χώρου.

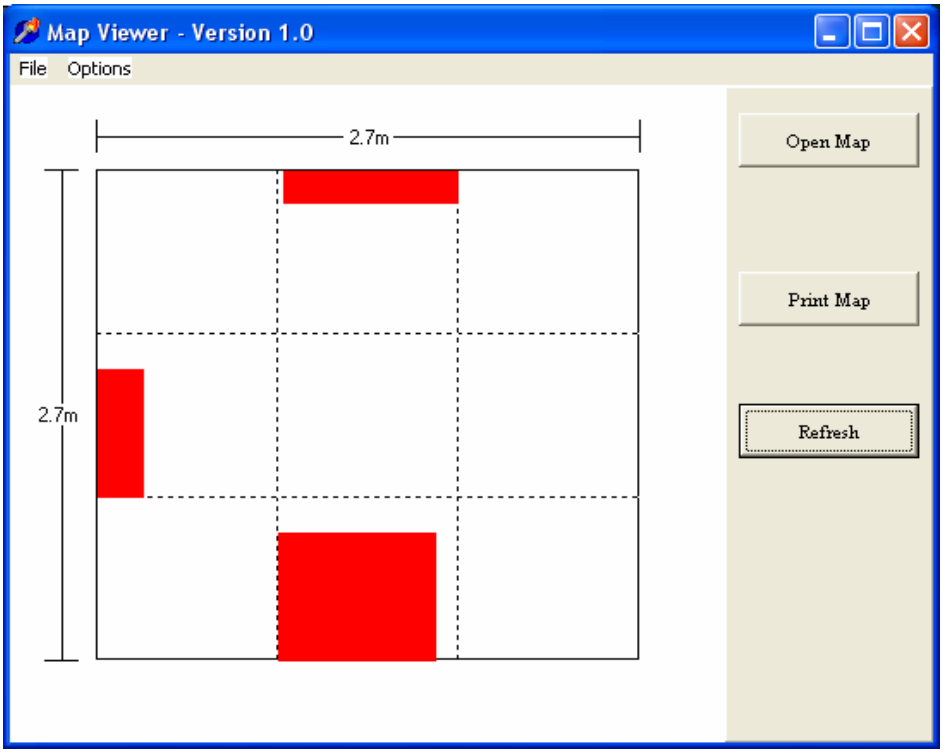
Συγκρίνοντας τις θέσεις και τις διαστάσεις των εμποδίων στο πραγματικό και στον χαρτογραφημένο κλειστό χώρο παρατηρούνται τα παρακάτω. Το εμπόδιο 1 έχει χαρτογραφηθεί σωστά. Επειδή οι γειτονικοί χώροι του εμποδίου 2 είναι προσπελάσιμοι από το ρομπότ η χαρτογράφηση του έγινε σωστά. Αντίθετα με τις δυο προηγούμενες περιπτώσεις το εμπόδιο3 δεν χαρτογραφείται σωστά. Αυτό οφείλεται στο γεγονός ότι στην πραγματικότητα το εμπόδιο βρίσκεται σε δυο υποχώρους.(1,4), (1,5) και ο υποχώρος (1,5) δεν μπορεί να προσπελαστεί από το ρομπότ για να μετρηθεί σωστά η απόσταση του εμποδίου από τον τοίχο. Ομοίως, το εμπόδιο 4 δεν χαρτογραφείται σωστά.

3.7 Χώρος με αντικείμενα διαστάσεων 2,7*2,7 m^2

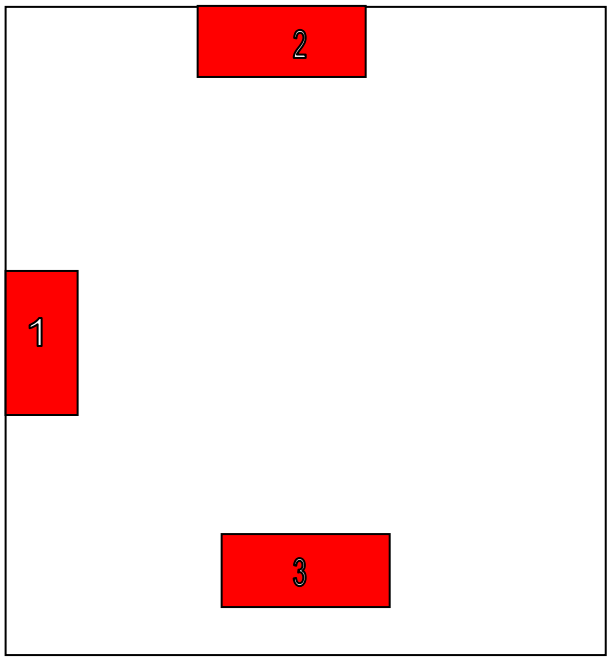
Τα αποτελέσματα που καταγράφηκαν μετά την πορεία που έκανε το ρομπότ παρουσιάζονται στο Πίνακα 3.3. Τα αποτελέσματα αυτά επεξεργάσθηκαν από το πρόγραμμα χαρτογράφησης στο οποίο γίνεται η τελική παρουσίαση του χώρου και φαίνεται στο Σχήμα 3.8. Τέλος στο Σχήμα 3.9 παρουσιάζεται σχηματικά ο πραγματικός χώρος.

δείκτης εμποδίου	απόσταση από την πάνω πλευρά του υποχώρου	απόσταση από την αριστερή πλευρά του υποχώρου	απόσταση από την δεξιά πλευρά του υποχώρου	απόσταση από την κάτω πλευρά του υποχώρου
0	2,00000	2,00000	2,00000	2,00000
1	2,00000	0,02612	2,00000	0,71752
0	2,00000	2,00000	2,00000	2,00000
1	0,19199	2,00000	0,66542	2,00000
0	2,00000	2,00000	2,00000	2,00000
0	2,00000	2,00000	2,00000	2,00000
0	2,00000	2,00000	2,00000	2,00000
1	0,19199	2,00000	0,11176	2,00000
0	2,00000	2,00000	2,00000	2,00000

Πίνακας 3.3: Παρουσίαση των αποτελεσμάτων χώρου με αντικείμενα, διαστάσεων 2,7*2,7 m^2.



Σχήμα 3.8: Παρουσίαση της μορφής που έχει ο προς εξέταση κλειστός χώρο



Σχήμα 3.9: Παρουσίαση του πραγματικού χώρου

Συγκρίνοντας τις θέσεις και τις διαστάσεις των εμποδίων στο πραγματικό και στον χαρτογραφημένο κλειστό χώρο παρατηρούνται τα παρακάτω. Το εμπόδιο 1 έχει χαρτογραφηθεί στη σωστή θέση αλλά φαίνεται να έχει μικρότερο μήκος από το κανονικο. Ομοίως το εμπόδιο 2 χαρτογραφήθηκε στο σωστό μέρος μέσα στο χώρο αλλά έχει διαφορά στο μέγεθός του. Το εμπόδιο 3 έχει χαρτογραφηθεί σωστά επειδή όμως δεν εφάπτεται στο τοίχο του κλειστού χώρου για αυτό στη χαρτογράφηση εμφανίζεται να έχει μεγαλύτερο μέγεθος.

Συμπεράσματα

Τα συμπεράσματα από την παρούσα διπλωματική εργασία μετά την δημιουργία και χαρτογράφηση των δωματίων είναι τα παρακάτω.

- Η χαρτογράφηση των χώρων ήταν ικανοποιητική.
- Για χώρους οι οποίοι δεν υπερβαίνουν τα 20 τετραγωνικά μέτρα το ρομπότ ανταποκρίνεται σωστά.
- Όταν οι προς χαρτογράφηση χώροι ξεπερνούν τα 20 τετραγωνικά μέτρα τότε αυξάνεται ο κίνδύνος το ρομπότ να βρεθεί εκτός της πορείας του. Επομένως αυξάνεται η πιθανόνητα λανθασμένης χαρτογράφησης.
- Θα πρέπει το πάτωμα του χώρου να είναι ομοιογενές δηλαδή από το ίδιο υλικό. Σε αντίθετη περίπτωση παρατηρήθηκε, ότι η κίνηση του ρομπότ έχει προβλήματα τα οποία οφείλονται σε σφάλματα που οφείλονται στην κίνηση των τροχών σε υλικό με διαφορετικό συντελεστή τριβής.

.

Μελλοντικές προεκτάσεις

- Αναλυτικότερη χαρτογράφηση με την χρήση της κάμερας.
- Φωτογράφηση των αντικειμένων που βρίσκονται μέσα στον προς εξέταση χώρο και ψηφιακή επεξεργασία της εικόνας, ώστε να υπάρχει δυνατότητα συλλογής περισσοτέρων δεδομένων.
- Ομαδοποίηση των αισθητήρων υπερήχου έτσι ώστε να είναι όσο το δυνατό πιο αντικειμενική η μέτρηση της απόστασης από τα αντικείμενα.
- Συνεργασία 2 η περισσοτέρων ρομπότ, επιτυγχάνοντας έτσι το ίδιο αποτέλεσμα σε λιγότερο χρόνο και επομένως μειώνεται ο χρόνος κίνησης των έντροχων οχημάτων. Έτσι μειώνονται τα λάθη τα οποία προκύπτουν και θα χαρτογραφούνται μεγαλύτεροι χώροι με μεγαλύτερη ακρίβεια.

ΚΩΔΙΚΕΣ

Κώδικας προσομοίωσης

```
//----------------- Header Definitions

#include "stdafx.h"
#include "stdlib.h"

#include <string.h>

/*
#include "userlib.h"
#include "mobilityutil.h"
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
*/

//----------------- Type Definitions

struct TSensor
{ float U_Dist;
  float D_Dist;
  float L_Dist;
  float R_Dist;
};

struct TFiles
{ char UF[20];
  char DF[20];
  char RF[20];
```

```
  char LF[20];
};

struct Node
{ bool Data;
  bool Gone;
  TFiles  Files;
  TSensor Dist;
};

//----------------- Global Variable Definitions

int NowI,NowJ,N,K;
int Robot_Up;

//mbyBasicRobot *m_pRobot;

TSensor Sensors;
TSensor MinDist;

Node *MAP;

Node *MAPX;

FILE *MAPF;

int CAM[4][4]; // Mapping for Use of Camera

//----------------- Other Functions
void TakeShot(char *S, int K) //K: 1=Up 2=Left 3=Right 4=Down
{}

float Sensor_Data(int K)
{
```

```
float F=0;
/*
F=sqrt((ranges.org.[K].x-ranges.end[K].x)*
      (ranges.org.[K].x-ranges.end[K].x)+
      (ranges.org.[K].y-ranges.end[K].y)*
      (ranges.org.[K].y-ranges.end[K].y));
*/

//--------------- ROOM SIMULATION
F=2;
switch(Robot_Up)
{

case 1:  if ((MAPX[N*(NowI-1)+NowJ].Data==true)&&((K== 0)||(K==23)))
                    { F=(float)0.40; }
                    if
((MAPX[N*(NowI+1)+NowJ].Data==true)&&((K==11)||(K==12)))
                    { F=(float)0.45; }
                    if ((MAPX[N*NowI+(NowJ-1)].Data==true)&&((K== 5)||(K== 6)))
                    { F=(float)0.50; }
                    if
((MAPX[N*NowI+(NowJ+1)].Data==true)&&((K==17)||(K==18)))
                    { F=(float)0.55; }
                    break;

case 2:  if ((MAPX[N*NowI+(NowJ-1)].Data==true)&&((K== 0)||(K==23)))
                    { F=(float)0.40; }
                    if
((MAPX[N*NowI+(NowJ+1)].Data==true)&&((K==11)||(K==12)))
                    { F=(float)0.45; }
                    if ((MAPX[N*(NowI+1)+NowJ].Data==true)&&((K== 5)||(K== 6)))
                    { F=(float)0.50; }
                    if                                          ((MAPX[N*(NowI-
1)+NowJ].Data==true)&&((K==17)||(K==18)))
```

```
                { F=(float)0.55; }
                break;

 case 3:  if ((MAPX[N*NowI+(NowJ+1)].Data==true)&&((K== 0)||(K==23)))
                { F=(float)0.40; }
                if                                          ((MAPX[N*NowI+(NowJ-
1)].Data==true)&&((K==11)||(K==12)))
                { F=(float)0.45; }
                if ((MAPX[N*(NowI-1)+NowJ].Data==true)&&((K== 5)||(K== 6)))
                { F=(float)0.50; }
                if
((MAPX[N*(NowI+1)+NowJ].Data==true)&&((K==17)||(K==18)))
                { F=(float)0.55; }
                break;

 case 4:  if ((MAPX[N*(NowI+1)+NowJ].Data==true)&&((K== 0)||(K==23)))
                { F=(float)0.40; }
                if                                          ((MAPX[N*(NowI-
1)+NowJ].Data==true)&&((K==11)||(K==12)))
                { F=(float)0.45; }
                if ((MAPX[N*NowI+(NowJ+1)].Data==true)&&((K== 5)||(K== 6)))
                { F=(float)0.50; }
                if                                          ((MAPX[N*NowI+(NowJ-
1)].Data==true)&&((K==17)||(K==18)))
                { F=(float)0.55; }
                break;
 }

 return(F);
}

void Read_Absolute_Sensors() // 1=Up 2=Left 3=Right 4=Down
{
//  m_pRobot->get_range_state(ranges);
```

```
switch(Robot_Up)
{

case 1:  Sensors.U_Dist=(Sensor_Data( 0)+Sensor_Data(23))/2;
        Sensors.D_Dist=(Sensor_Data(11)+Sensor_Data(12))/2;
        Sensors.L_Dist=(Sensor_Data( 5)+Sensor_Data( 6))/2;
        Sensors.R_Dist=(Sensor_Data(17)+Sensor_Data(18))/2;
                MinDist.U_Dist=(float)1.0475;
                MinDist.D_Dist=(float)1.0475;
                MinDist.L_Dist=(float)1.0835;
                MinDist.R_Dist=(float)1.0835;
        break;

case 2:  Sensors.R_Dist=(Sensor_Data( 0)+Sensor_Data(23))/2;
        Sensors.L_Dist=(Sensor_Data(11)+Sensor_Data(12))/2;
        Sensors.U_Dist=(Sensor_Data( 5)+Sensor_Data( 6))/2;
        Sensors.D_Dist=(Sensor_Data(17)+Sensor_Data(18))/2;
                MinDist.U_Dist=(float)1.0835;
                MinDist.D_Dist=(float)1.0835;
                MinDist.L_Dist=(float)1.0475;
                MinDist.R_Dist=(float)1.0475;
        break;

case 3:  Sensors.L_Dist=(Sensor_Data( 0)+Sensor_Data(23))/2;
        Sensors.R_Dist=(Sensor_Data(11)+Sensor_Data(12))/2;
        Sensors.D_Dist=(Sensor_Data( 5)+Sensor_Data( 6))/2;
        Sensors.U_Dist=(Sensor_Data(17)+Sensor_Data(18))/2;
                MinDist.U_Dist=(float)1.0835;
                MinDist.D_Dist=(float)1.0835;
                MinDist.L_Dist=(float)1.0475;
                MinDist.R_Dist=(float)1.0475;
        break;

case 4:  Sensors.D_Dist=(Sensor_Data( 0)+Sensor_Data(23))/2;
```

```
        Sensors.U_Dist=(Sensor_Data(11)+Sensor_Data(12))/2;
        Sensors.R_Dist=(Sensor_Data( 5)+Sensor_Data( 6))/2;
        Sensors.L_Dist=(Sensor_Data(17)+Sensor_Data(18))/2;
                MinDist.U_Dist=(float)1.0475;
                MinDist.D_Dist=(float)1.0475;
                MinDist.L_Dist=(float)1.0835;
                MinDist.R_Dist=(float)1.0835;
        break;
  }
}

//-----------------

bool CanGo(int Dir)  // 1=Up 2=Left 3=Right 4=Down
{
         int I,J;
         char S[20];
         char SI[5],SJ[5];

         Read_Absolute_Sensors();

         //------- Up Check

   if (((MAP[N*(NowI-1)+NowJ].Data==false)&&
              (Sensors.U_Dist<=MinDist.U_Dist)) ||
      ((MAP[N*(NowI-1)+NowJ].Data==true)&&
              (MAP[N*(NowI-1)+NowJ].Dist.D_Dist>Sensors.U_Dist-
(MinDist.U_Dist-0.9))))
         { MAP[N*(NowI-1)+NowJ].Data=true;
           MAP[N*(NowI-1)+NowJ].Dist.D_Dist=(float)(Sensors.U_Dist-
(MinDist.U_Dist-0.9));
           //---------
           I=NowI-1; J=NowJ;
           itoa(I,SI,10);
```

```
          itoa(J,SJ,10);
          strcpy(S,SI);
          strcat(S,"x");
          strcat(S,SJ);
          strcat(S,"_1.jpg");
          strcpy(MAP[N*(NowI-1)+NowJ].Files.DF,S);
          TakeShot(S,CAM[0][Robot_Up-1]);
         }

         //------- Left Check

  if (((MAP[N*NowI+(NowJ-1)].Data==false)&&
            (Sensors.L_Dist<=MinDist.L_Dist)) ||
     ((MAP[N*NowI+(NowJ-1)].Data==true)&&
            (MAP[N*NowI+(NowJ-1)].Dist.R_Dist>Sensors.L_Dist-(MinDist.L_Dist-
0.9))))
         { MAP[N*NowI+(NowJ-1)].Data=true;
          MAP[N*NowI+(NowJ-1)].Dist.R_Dist=(float)(Sensors.L_Dist-
(MinDist.L_Dist-0.9));
          //---------
          I=NowI; J=NowJ-1;
          itoa(I,SI,10);
          itoa(J,SJ,10);
          strcpy(S,SI);
          strcat(S,"x");
          strcat(S,SJ);
          strcat(S,"_2.jpg");
          strcpy(MAP[N*NowI+(NowJ-1)].Files.RF,S);
          TakeShot(S,CAM[1][Robot_Up-1]);
         }

         //------- Right Check

  if (((MAP[N*NowI+(NowJ+1)].Data==false)&&
```

```
            (Sensors.R_Dist<=MinDist.R_Dist)) ||
       ((MAP[N*NowI+(NowJ+1)].Data==true)&&
            (MAP[N*NowI+(NowJ+1)].Dist.L_Dist>Sensors.R_Dist-
(MinDist.R_Dist-0.9))))
          { MAP[N*NowI+(NowJ+1)].Data=true;
           MAP[N*NowI+(NowJ+1)].Dist.L_Dist=(float)(Sensors.R_Dist-
(MinDist.R_Dist-0.9));
           //---------
           I=NowI; J=NowJ+1;
           itoa(I,SI,10);
           itoa(J,SJ,10);
           strcpy(S,SI);
           strcat(S,"x");
           strcat(S,SJ);
           strcat(S,"_3.jpg");
           strcpy(MAP[N*NowI+(NowJ+1)].Files.LF,S);
           TakeShot(S,CAM[2][Robot_Up-1]);
          }

          //------- Down Check

   if (((MAP[N*(NowI+1)+NowJ].Data==false)&&
            (Sensors.D_Dist<=MinDist.D_Dist)) ||
       ((MAP[N*(NowI+1)+NowJ].Data==true)&&
            (MAP[N*(NowI+1)+NowJ].Dist.U_Dist>Sensors.D_Dist-
(MinDist.D_Dist-0.9))))
          { MAP[N*(NowI+1)+NowJ].Data=true;
           MAP[N*(NowI+1)+NowJ].Dist.U_Dist=(float)(Sensors.D_Dist-
(MinDist.D_Dist-0.9));
           //---------
           I=NowI+1; J=NowJ;
           itoa(I,SI,10);
           itoa(J,SJ,10);
           strcpy(S,SI);
```

```
            strcat(S,"x");
            strcat(S,SJ);
            strcat(S,"_4.jpg");
            strcpy(MAP[N*(NowI+1)+NowJ].Files.UF,S);
            TakeShot(S,CAM[3][Robot_Up-1]);
          }

          //----------------------

          switch(Dir)
          {case 1: if ((MAP[N*(NowI-1)+NowJ].Data==false)&&
              (MAP[N*(NowI-1)+NowJ].Gone==false))
                    {return(true);}
                else {return(false);}
           break;
      case 2: if ((MAP[N*NowI+(NowJ-1)].Data==false)&&
                                    (MAP[N*NowI+(NowJ-1)].Gone==false))
                               {return(true);}
           else {return(false);}
          break;
      case 3: if ((MAP[N*NowI+(NowJ+1)].Data==false)&&
              (MAP[N*NowI+(NowJ+1)].Gone==false))
                               {return(true);}
           else {return(false);}
          break;
      case 4: if ((MAP[N*(NowI+1)+NowJ].Data==false)&&
                                    (MAP[N*(NowI+1)+NowJ].Gone==false))
                               {return(true);}
           else {return(false);}
                        break;
          default: return(false);
                         break;
          }
}
```

```
//-----------------

void Do_Move(int OldI,int OldJ,int NewI,int NewJ)
{   /*
   //-------Move Up
         if ( ((Robot_Up==1)&&(NewI<OldI)&&(NewJ=OldJ))||
              ((Robot_Up==2)&&(NewI=OldI)&&(NewJ<OldJ))||
              ((Robot_Up==3)&&(NewI=OldI)&&(NewJ>OldJ))||
                   ((Robot_Up==4)&&(NewI>OldI)&&(NewJ=OldJ)) )
         { m_pRobot->send_velocity_command(0.3,0.0);
           omni_thread::sleep(4.7);
           m_pRobot->send_velocity_command(0.0,0.0);
     omni_thread::sleep(2.0);
         }

   //------- Move Left
   if ( ((Robot_Up==1)&&(NewI=OldI)&&(NewJ<OldJ))||
              ((Robot_Up==2)&&(NewI>OldI)&&(NewJ=OldJ))||
              ((Robot_Up==3)&&(NewI<OldI)&&(NewJ=OldJ))||
                   ((Robot_Up==4)&&(NewI=OldI)&&(NewJ>OldJ)) )
         { m_pRobot->send_velocity_command(0.0,1.0);
           omni_thread::sleep(2.0);
           m_pRobot->send_velocity_command(0.0,0.0);
     m_pRobot->send_velocity_command(0.3,0.0);
           omni_thread::sleep(4.7);
           m_pRobot->send_velocity_command(0.0,0.0);
     omni_thread::sleep(2.0);
         }

   //------- Move Right
   if ( ((Robot_Up==1)&&(NewI=OldI)&&(NewJ>OldJ))||
              ((Robot_Up==2)&&(NewI<OldI)&&(NewJ=OldJ))||
              ((Robot_Up==3)&&(NewI>OldI)&&(NewJ=OldJ))||
```

```
                ((Robot_Up==4)&&(NewI=OldI)&&(NewJ<OldJ)) )
        { m_pRobot->send_velocity_command(0.0,-1.0);
         omni_thread::sleep(2.0);
         m_pRobot->send_velocity_command(0.0,0.0);
    m_pRobot->send_velocity_command(0.3,0.0);
         omni_thread::sleep(4.7);
         m_pRobot->send_velocity_command(0.0,0.0);
    omni_thread::sleep(2.0);
        }

   //-------Move Down
   if ( ((Robot_Up==1)&&(NewI>OldI)&&(NewJ=OldJ))||
          ((Robot_Up==2)&&(NewI=OldI)&&(NewJ>OldJ))||
          ((Robot_Up==3)&&(NewI=OldI)&&(NewJ<OldJ))||
                ((Robot_Up==4)&&(NewI<OldI)&&(NewJ=OldJ)) )
        { m_pRobot->send_velocity_command(-0.3,0.0);
         omni_thread::sleep(4.7);
         m_pRobot->send_velocity_command(0.0,0.0);
    omni_thread::sleep(2.0);
        }
        */
}

//-----------------

void Move(int I,int J) // 1=Up 2=Left 3=Right 4=Down
{  MAP[N*I+J].Gone=true;
  //-------------------
  if ((I!=NowI)||(J!=NowJ)) { Do_Move(NowI,NowJ,I,J);
                    NowI=I;
                                                NowJ=J;}
  //-------------------
  if (CanGo(1)&&(I>0)) { Do_Move(NowI,NowJ,I-1,J);
                  NowI=I-1;
```

```
                                        NowJ=J;
                    Move(I-1,J);}
  //-------------------
  if ((I!=NowI)||(J!=NowJ)) { Do_Move(NowI,NowJ,I,J);
                        NowI=I;
                                                        NowJ=J;}
  //-------------------
  if (CanGo(4)&&(I<N-1)) { Do_Move(NowI,NowJ,I+1,J);
                      NowI=I+1;
                                         NowJ=J;
                                         Move(I+1,J);}
  //-------------------
  if ((I!=NowI)||(J!=NowJ)) { Do_Move(NowI,NowJ,I,J);
                        NowI=I;
                                                        NowJ=J;}
  //-------------------
  if (CanGo(2)&&(J>0)) { Do_Move(NowI,NowJ,I,J-1);
                    NowI=I;
                                        NowJ=J-1;
                                        Move(I,J-1);}
  //-------------------
  if ((I!=NowI)||(J!=NowJ)) { Do_Move(NowI,NowJ,I,J);
                        NowI=I;
                                                        NowJ=J;}
  //-------------------
  if (CanGo(3)&&(J<N-1)) { Do_Move(NowI,NowJ,I,J+1);
                      NowI=I;
                                         NowJ=J+1;
                                         Move(I,J+1);}
  //-------------------
  if ((I!=NowI)||(J!=NowJ)) { Do_Move(NowI,NowJ,I,J);
                        NowI=I;
                                                        NowJ=J;}
}
```

```
//----------------- MAIN

void main()
{

 //-------------------- Construction Code

 CAM[0][0]=1;CAM[1][0]=2;CAM[2][0]=3;CAM[3][0]=4;
 CAM[0][1]=3;CAM[1][1]=1;CAM[2][1]=4;CAM[3][1]=2;
 CAM[0][2]=2;CAM[1][2]=4;CAM[2][2]=1;CAM[3][2]=3;
 CAM[0][3]=4;CAM[1][3]=3;CAM[2][3]=2;CAM[3][3]=1;



 /*
 int argc=3;
 char* argv[3];
 argv[0]="program";
 argv[1]="-robot";
 argv[2]="ATRVMini";

 char *modulename;
 char *robotname;

 SystemModule_i::init_orb(argc,argv);
 modulename=mbyUtility::get_option(argc,argv,"-name","UserTest");
 mbyBasicModule *m_pModule=new mbyBasicModule(modulename,argc,argv);
 robotname=mbyUtility::get_option(argc,argv,"-robot",ATRVMini");
 m_pRobot=new mbyBasicRobot("RobotX",robotname);

 if(m_pModule->add_new_component(m_pRobot)<0) exit(-1);
 m_pModule->start_module();

 MobilityGeometry::SegmentData ranges;
```

```
MobilityActuator::ActuatorData OurCommand;
*/
//--------------------

int i,j;

Robot_Up=1; // Up for Robot = Up for Map

printf("Give N(m):"); scanf("%d",&N);

K=(int) (N/(0.9));
if ((N-0.9*K)>0) { K+=1; }
N=K;
printf("N=%d\n",N);

//----------------- MAPX Construction  (Room Simulation)
MAPX=(Node *)malloc(N*N*sizeof(Node));
for (i=0;i<N;i++)
 for (j=0;j<N;j++)
 { MAPX[N*i+j].Data=false; }
//------
 MAPX[N*0+0].Data=true;
 MAPX[N*0+5].Data=true;
 MAPX[N*0+7].Data=true;
 MAPX[N*1+3].Data=true;
 MAPX[N*1+4].Data=true;
 MAPX[N*1+5].Data=true;
 MAPX[N*1+7].Data=true;
 MAPX[N*2+7].Data=true;
 MAPX[N*2+8].Data=true;
 MAPX[N*3+1].Data=true;
 MAPX[N*4+1].Data=true;
 MAPX[N*4+6].Data=true;
 MAPX[N*4+8].Data=true;
```

```
 MAPX[N*5+0].Data=true;
 MAPX[N*5+1].Data=true;
 MAPX[N*5+6].Data=true;
 MAPX[N*5+8].Data=true;
 MAPX[N*6+6].Data=true;
 MAPX[N*7+2].Data=true;
 MAPX[N*7+4].Data=true;
 MAPX[N*7+5].Data=true;
 MAPX[N*7+8].Data=true;
 MAPX[N*8+1].Data=true;
 MAPX[N*8+2].Data=true;
 MAPX[N*8+7].Data=true;
 MAPX[N*8+8].Data=true;
//----------------- MAP Memory Allocation & Initialization
MAP=(Node *)malloc(N*N*sizeof(Node));
for (i=0;i<N;i++)
 for (j=0;j<N;j++)
 { MAP[N*i+j].Data=false;
   MAP[N*i+j].Gone=false;
   MAP[N*i+j].Dist.U_Dist=2;
         MAP[N*i+j].Dist.L_Dist=2;
         MAP[N*i+j].Dist.R_Dist=2;
         MAP[N*i+j].Dist.D_Dist=2;
   strcpy(MAP[N*i+j].Files.UF,"");
   strcpy(MAP[N*i+j].Files.LF,"");
   strcpy(MAP[N*i+j].Files.RF,"");
   strcpy(MAP[N*i+j].Files.DF,"");
 }
NowI=(int) (N/2);
NowJ=(int) (N/2);
//----------------- Calculate & Save MAP
Move(NowI,NowJ);
//----------------- Open Save & Close File
MAPF = fopen("MAP.txt", "w");
```

```
  fprintf(MAPF,"%d\n",N);
  for (i=0;i<N;i++)
   for (j=0;j<N;j++)
   { fprintf(MAPF,"%d %.5f %.5f %.5f %.5f\n",MAP[N*i+j].Data,
                                              MAP[N*i+j].Dist.U_Dist,

MAP[N*i+j].Dist.L_Dist,

MAP[N*i+j].Dist.R_Dist,

MAP[N*i+j].Dist.D_Dist);
            fprintf(MAPF,"%s\n",MAP[N*i+j].Files.UF);
            fprintf(MAPF,"%s\n",MAP[N*i+j].Files.LF);
            fprintf(MAPF,"%s\n",MAP[N*i+j].Files.RF);
            fprintf(MAPF,"%s\n",MAP[N*i+j].Files.DF);
   }
  fclose(MAPF);
  //---------------------- Destruction Code
  /*
  delete m_pModule;
  delete m_pRobot;
  */
  delete MAP;
  delete MAPX;
}
```

Κώδικας εργαστηρίου κυρίως

```
#include <string.h>
#include "userlib.h"
#include "mobilityutil.h"
#include <stdio.h>
#include <stdlib.h>
#include <math.h>


//----------------- Type Definitions

struct TSensor
{ float U_Dist;
  float D_Dist;
  float L_Dist;
  float R_Dist;
};

struct TFiles
{ char UF[20];
  char DF[20];
  char RF[20];
  char LF[20];
};

struct Node
{ bool Data;
  bool Gone;
  TFiles  Files;
  TSensor Dist;
};

//----------------- Global Variable Definitions
```

```
int NowI,NowJ,N,K;
int Robot_Up;

mbyBasicModule *m_pModule;
mbyBasicRobot *m_pRobot;

TSensor Sensors;
TSensor MinDist;

Node *MAP;

Node *MAPX;

FILE *MAPF;

int CAM[4][4]; // Mapping for Use of Camera

//----------------- Other Functions
void TakeShot(char *S, int K) //K: 1=Up 2=Left 3=Right 4=Down
{}

float min(float A,float B)
{ float f;
   if (A<B) {f=A;}
      else  {f=B;}
    return (f) ;
}

float Sensor_Data(int K)
{
 float F=0;

 F=sqrt((ranges.org.[K].x-ranges.end[K].x)*
       (ranges.org.[K].x-ranges.end[K].x)+
```

```
        (ranges.org.[K].y-ranges.end[K].y)*
        (ranges.org.[K].y-ranges.end[K].y));

         return (f) ;
}



}

void Read_Absolute_Sensors() // 1=Up 2=Left 3=Right 4=Down
{
 m_pRobot->get_range_state(ranges);
 switch(Robot_Up)
 {

 case 1:  Sensors.U_Dist=(Sensor_Data( 0),Sensor_Data(23))/2;
         Sensors.D_Dist=(Sensor_Data(11),Sensor_Data(12))/2;
         Sensors.L_Dist=(Sensor_Data( 5),Sensor_Data( 6))/2;
         Sensors.R_Dist=(Sensor_Data(17),Sensor_Data(18))/2;
                  MinDist.U_Dist=(float)1.0475;
                  MinDist.D_Dist=(float)1.0475;
                  MinDist.L_Dist=(float)1.0835;
                  MinDist.R_Dist=(float)1.0835;
         break;

 case 2:  Sensors.R_Dist=(Sensor_Data( 0),Sensor_Data(23))/2;
         Sensors.L_Dist=(Sensor_Data(11),Sensor_Data(12))/2;
         Sensors.U_Dist=(Sensor_Data( 5),Sensor_Data( 6))/2;
         Sensors.D_Dist=(Sensor_Data(17),Sensor_Data(18))/2;
                  MinDist.U_Dist=(float)1.0835;
                  MinDist.D_Dist=(float)1.0835;
                  MinDist.L_Dist=(float)1.0475;
                  MinDist.R_Dist=(float)1.0475;
         break;
```

```
 case 3:  Sensors.L_Dist=(Sensor_Data( 0),Sensor_Data(23))/2;
        Sensors.R_Dist=(Sensor_Data(11),Sensor_Data(12))/2;
        Sensors.D_Dist=(Sensor_Data( 5),Sensor_Data( 6))/2;
        Sensors.U_Dist=(Sensor_Data(17),Sensor_Data(18))/2;
                MinDist.U_Dist=(float)1.0835;
                MinDist.D_Dist=(float)1.0835;
                MinDist.L_Dist=(float)1.0475;
                MinDist.R_Dist=(float)1.0475;
        break;

 case 4:  Sensors.D_Dist=(Sensor_Data( 0),Sensor_Data(23))/2;
        Sensors.U_Dist=(Sensor_Data(11),Sensor_Data(12))/2;
        Sensors.R_Dist=(Sensor_Data( 5),Sensor_Data( 6))/2;
        Sensors.L_Dist=(Sensor_Data(17),Sensor_Data(18))/2;
                MinDist.U_Dist=(float)1.0475;
                MinDist.D_Dist=(float)1.0475;
                MinDist.L_Dist=(float)1.0835;
                MinDist.R_Dist=(float)1.0835;
        break;
 }
}

//-----------------

bool CanGo(int Dir)  // 1=Up 2=Left 3=Right 4=Down
{

         char S[20];


         Read_Absolute_Sensors();

         //------- Up Check
```

```
   if (((MAP[N*(NowI-1)+NowJ].Data==false)&&
            (Sensors.U_Dist<=MinDist.U_Dist)) ||
      ((MAP[N*(NowI-1)+NowJ].Data==true)&&
            (MAP[N*(NowI-1)+NowJ].Dist.D_Dist>Sensors.U_Dist-
(MinDist.U_Dist-0.9))))
         { MAP[N*(NowI-1)+NowJ].Data=true;
          MAP[N*(NowI-1)+NowJ].Dist.D_Dist=(float)(Sensors.U_Dist-
(MinDist.U_Dist-0.9));
          //---------

          strcpy(S,SI);
          strcpy(MAP[N*(NowI-1)+NowJ].Files.DF,S);
          TakeShot(S,CAM[0][Robot_Up-1]);
         }

        //------- Left Check

   if (((MAP[N*NowI+(NowJ-1)].Data==false)&&
            (Sensors.L_Dist<=MinDist.L_Dist)) ||
      ((MAP[N*NowI+(NowJ-1)].Data==true)&&
            (MAP[N*NowI+(NowJ-1)].Dist.R_Dist>Sensors.L_Dist-(MinDist.L_Dist-
0.9))))
         { MAP[N*NowI+(NowJ-1)].Data=true;
          MAP[N*NowI+(NowJ-1)].Dist.R_Dist=(float)(Sensors.L_Dist-
(MinDist.L_Dist-0.9));
          //---------

          strcpy(S,SI);
          strcpy(MAP[N*NowI+(NowJ-1)].Files.RF,S);
          TakeShot(S,CAM[1][Robot_Up-1]);
         }

        //------- Right Check
```

```
   if (((MAP[N*NowI+(NowJ+1)].Data==false)&&
            (Sensors.R_Dist<=MinDist.R_Dist)) ||
      ((MAP[N*NowI+(NowJ+1)].Data==true)&&
            (MAP[N*NowI+(NowJ+1)].Dist.L_Dist>Sensors.R_Dist-
(MinDist.R_Dist-0.9))))
         { MAP[N*NowI+(NowJ+1)].Data=true;
          MAP[N*NowI+(NowJ+1)].Dist.L_Dist=(float)(Sensors.R_Dist-
(MinDist.R_Dist-0.9));
          //---------

          strcpy(S,SI);
          strcpy(MAP[N*NowI+(NowJ+1)].Files.LF,S);
          TakeShot(S,CAM[2][Robot_Up-1]);
         }

        //------- Down Check

   if (((MAP[N*(NowI+1)+NowJ].Data==false)&&
            (Sensors.D_Dist<=MinDist.D_Dist)) ||
      ((MAP[N*(NowI+1)+NowJ].Data==true)&&
            (MAP[N*(NowI+1)+NowJ].Dist.U_Dist>Sensors.D_Dist-
(MinDist.D_Dist-0.9))))
         { MAP[N*(NowI+1)+NowJ].Data=true;
          MAP[N*(NowI+1)+NowJ].Dist.U_Dist=(float)(Sensors.D_Dist-
(MinDist.D_Dist-0.9));
          //---------

          strcpy(S,SI);
          strcpy(MAP[N*(NowI+1)+NowJ].Files.UF,S);
          TakeShot(S,CAM[3][Robot_Up-1]);
         }

        //-----------------------
```

```
          switch(Dir)
          {case 1: if ((MAP[N*(NowI-1)+NowJ].Data==false)&&
              (MAP[N*(NowI-1)+NowJ].Gone==false))
                    {return(true);}
                 else {return(false);}
           break;
        case 2: if ((MAP[N*NowI+(NowJ-1)].Data==false)&&
                   (MAP[N*NowI+(NowJ-1)].Gone==false))
                    {return(true);}
           else {return(false);}
          break;
        case 3: if ((MAP[N*NowI+(NowJ+1)].Data==false)&&
              (MAP[N*NowI+(NowJ+1)].Gone==false))
                   {return(true);}
           else {return(false);}
          break;
         case 4: if ((MAP[N*(NowI+1)+NowJ].Data==false)&&
                    (MAP[N*(NowI+1)+NowJ].Gone==false))
                   {return(true);}
           else {return(false);}
                     break;
          default: return(false);
                     break;
          }
}

//-----------------

void Do_Move(int OldI,int OldJ,int NewI,int NewJ)
{
    printf("[%d,%d]                    ->                    [%d,%d]
:CompassMode=[%d]\n",OldI,OldJ,NewI,NewJ,Robot_Up);
```

```
//-------Move Up
   if ( ((Robot_Up==1)&&(NewI<OldI)&&(NewJ==OldJ))||
         ((Robot_Up==2)&&(NewI==OldI)&&(NewJ<OldJ))||
         ((Robot_Up==3)&&(NewI==OldI)&&(NewJ>OldJ))||
       ((Robot_Up==4)&&(NewI>OldI)&&(NewJ==OldJ)) )
     {printf("want to go Up :");
        m_pRobot->send_velocity_command(0.3,0.0);
        omni_thread::sleep(4.7);
        m_pRobot->send_velocity_command(0.0,0.0);
     omni_thread::sleep(2.0);
     printf("Moved up\n");
       }

//------- Move Left
if    ( ((Robot_Up==1)&&(NewI==OldI)&&(NewJ<OldJ))||
         ((Robot_Up==2)&&(NewI>OldI)&&(NewJ==OldJ))||
         ((Robot_Up==3)&&(NewI<OldI)&&(NewJ==OldJ))||
          ((Robot_Up==4)&&(NewI=OldI)&&(NewJ>OldJ)) )
      { printf("want to go Left :");
    m_pRobot->send_velocity_command(0.0,1.0);
        omni_thread::sleep(2.0);
        m_pRobot->send_velocity_command(0.0,0.0);
  m_pRobot->send_velocity_command(0.3,0.0);
        omni_thread::sleep(4.7);
        m_pRobot->send_velocity_command(0.0,0.0);
  omni_thread::sleep(2.0);
       printf("Moved left\n");
}

//------- Move Right
if    ( ((Robot_Up==1)&&(NewI==OldI)&&(NewJ>OldJ))||
         ((Robot_Up==2)&&(NewI<OldI)&&(NewJ==OldJ))||
         ((Robot_Up==3)&&(NewI>OldI)&&(NewJ==OldJ))||
         ((Robot_Up==4)&&(NewI==OldI)&&(NewJ<OldJ)) )
```

```
          {printf("want to go right :");
        m_pRobot->send_velocity_command(0.0,-1.0);
          omni_thread::sleep(2.0);
          m_pRobot->send_velocity_command(0.0,0.0);
    m_pRobot->send_velocity_command(0.3,0.0);
          omni_thread::sleep(4.7);
          m_pRobot->send_velocity_command(0.0,0.0);
    omni_thread::sleep(2.0);
         printf("Moved right\n");
  }

   //-------Move Down
   if    ( ((Robot_Up==1)&&(NewI>OldI)&&(NewJ==OldJ))||
           ((Robot_Up==2)&&(NewI==OldI)&&(NewJ>OldJ))||
           ((Robot_Up==3)&&(NewI=OldI)&&(NewJ<OldJ))||
         ((Robot_Up==4)&&(NewI<OldI)&&(NewJ==OldJ)) )
         {printf("want to go Down :");
        m_pRobot->send_velocity_command(-0.3,0.0);
          omni_thread::sleep(4.7);
          m_pRobot->send_velocity_command(0.0,0.0);
    omni_thread::sleep(2.0);
         printf("Moved down\n");}

}

//-----------------

void Move(int I,int J) // 1=Up 2=Left 3=Right 4=Down
{  MAP[N*I+J].Gone=true;
  //-------------------
  if ((I!=NowI)||(J!=NowJ)) { Do_Move(NowI,NowJ,I,J);
                        NowI=I;
                                                    NowJ=J;}
  //-------------------
```

```
if (CanGo(1)&&(I>0)) { Do_Move(NowI,NowJ,I-1,J);
                NowI=I-1;
                                NowJ=J;
                Move(I-1,J);}
//-------------------
if ((I!=NowI)||(J!=NowJ)) { Do_Move(NowI,NowJ,I,J);
                  NowI=I;
                                                NowJ=J;}
//-------------------
if (CanGo(4)&&(I<N-1)) { Do_Move(NowI,NowJ,I+1,J);
                 NowI=I+1;
                                 NowJ=J;
                                 Move(I+1,J);}
//-------------------
if ((I!=NowI)||(J!=NowJ)) { Do_Move(NowI,NowJ,I,J);
                  NowI=I;
                                                NowJ=J;}
//-------------------
if (CanGo(2)&&(J>0)) { Do_Move(NowI,NowJ,I,J-1);
                NowI=I;
                                NowJ=J-1;
                                Move(I,J-1);}
//-------------------
if ((I!=NowI)||(J!=NowJ)) { Do_Move(NowI,NowJ,I,J);
                  NowI=I;
                                                NowJ=J;}
//-------------------
if (CanGo(3)&&(J<N-1)) { Do_Move(NowI,NowJ,I,J+1);
                 NowI=I;
                                 NowJ=J+1;
                                 Move(I,J+1);}
//-------------------
if ((I!=NowI)||(J!=NowJ)) { Do_Move(NowI,NowJ,I,J);
                  NowI=I;
```

```
                                                    NowJ=J;}
}

//----------------- MAIN

void main()
{

  //-------------------- Construction Code

  CAM[0][0]=1;CAM[1][0]=2;CAM[2][0]=3;CAM[3][0]=4;
  CAM[0][1]=3;CAM[1][1]=1;CAM[2][1]=4;CAM[3][1]=2;
  CAM[0][2]=2;CAM[1][2]=4;CAM[2][2]=1;CAM[3][2]=3;
  CAM[0][3]=4;CAM[1][3]=3;CAM[2][3]=2;CAM[3][3]=1;

  L[0]=2;R[0]=3;
  L[1]=4;R[1]=1;
  L[2]=1;R[2]=4;
  L[3]=3;R[3]=2;

  int argc=3;
  char* argv[3];
  argv[0]="program";
  argv[1]="-robot";
  argv[2]="ATRVMini";

  char *modulename;
  char *robotname;

  SystemModule_i::init_orb(argc,argv);
  modulename=mbyUtility::get_option(argc,argv,"-name","UserTest");
  mbyBasicModule *m_pModule=new mbyBasicModule(modulename,argc,argv);
  robotname=mbyUtility::get_option(argc,argv,"-robot",ATRVMini");
  m_pRobot=new mbyBasicRobot("RobotX",robotname);
```

```
if(m_pModule->add_new_component(m_pRobot)<0) exit(-1);
m_pModule->start_module();

MobilityGeometry::SegmentData ranges;
MobilityActuator::ActuatorData OurCommand;

//--------------------

int i,j;

Robot_Up=1; // Up for Robot = Up for Map

printf("Give N(m):"); scanf("%d",&N);

K=(int) (N/(0.9));
if ((N-0.9*K)>0) { K+=1; }
N=K;
printf("MAP SIZE =(%dX%D)\n",N,N);



//----------------- MAP Memory Allocation & Initialization
MAP=(Node *)malloc(N*N*sizeof(Node));
for (i=0;i<N;i++)
 for (j=0;j<N;j++)
 {    MAP[N*i+j].Data=false;
      MAP[N*i+j].Gone=false;
      MAP[N*i+j].Dist.U_Dist=2;
         MAP[N*i+j].Dist.L_Dist=2;
         MAP[N*i+j].Dist.R_Dist=2;
         MAP[N*i+j].Dist.D_Dist=2;
  strcpy(MAP[N*i+j].Files.UF,"");
  strcpy(MAP[N*i+j].Files.LF,"");
  strcpy(MAP[N*i+j].Files.RF,"");
```

```
   strcpy(MAP[N*i+j].Files.DF,"");
  }
 NowI=(int) (N/2);
 NowJ=(int) (N/2);
  printf (start [i,j]=[%d,%d]\n"NowI,NowJ);

 //----------------- Calculate & Save MAP
 Move(NowI,NowJ);
 //----------------- Open Save & Close File
 MAPF = fopen("MAP.txt", "w");

 fprintf(MAPF,"%d\n",N);
 for (i=0;i<N;i++)
  for (j=0;j<N;j++)
  { fprintf(MAPF,"%d %.5f %.5f %.5f %.5f\n",MAP[N*i+j].Data,
                                        MAP[N*i+j].Dist.U_Dist,
                                                MAP[N*i+j].Dist.L_Dist,
                                                MAP[N*i+j].Dist.R_Dist,
                                                MAP[N*i+j].Dist.D_Dist);
          fprintf(MAPF,"%s\n",MAP[N*i+j].Files.UF);
          fprintf(MAPF,"%s\n",MAP[N*i+j].Files.LF);
          fprintf(MAPF,"%s\n",MAP[N*i+j].Files.RF);
          fprintf(MAPF,"%s\n",MAP[N*i+j].Files.DF);
  }
 fclose(MAPF);
 //-------------------- Destruction Code
 /*
 delete m_pModule;
 delete m_pRobot;
 */
 delete MAP;

}
```

Κώδικας καταγραφής πορείας

```
#include "userlib.h"
#include "mobilityutil.h"
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc,char *argv[])
{
char *modulename;
char *robotname;

//int argc=3;
//char* argv[3];
//argv[0]="navigdata";
//argv[1]="-robot";
//argv[2]="ATRVMini";


//char *modulename;
//char *robotbame;

float x,y,theta;

FILE *NAV;
FILE *sfp;
//mbyBasicModule *m_pModule;
//mbyBasicRobot *m_pRobot;


SystemModule_i::init_orb(argc,argv);
modulename=mbyUtility::get_option(argc,argv,"-name","UserTest");
mbyBasicModule  *m_pModule=new mbyBasicModule(modulename,argc,argv);
robotname=mbyUtility::get_option(argc,argv,"-robot","ATRVMini");
mbyBasicRobot  *m_pRobot=new mbyBasicRobot("RobotX",robotname);


if(m_pModule->add_new_component(m_pRobot)<0) exit(-1);
m_pModule->start_module();

MobilityGeometry::SegmentData ranges;
MobilityActuator::ActuatorData OurCommand;

NAV=fopen("poria11.dat", "w+");

while(1)
 {
     omni_thread::sleep(0,100000000);
     m_pRobot->get_drive_state(x,y,theta);
//     omni_thread::sleep(0.1);
     fprintf(NAV,"X=%f Y=%f\n" ,x,y);
//     omni_thread::sleep(0.1);
     sfp=fopen("stop.txt","r");
      if (sfp==NULL)

      {
      delete m_pModule;
      delete m_pRobot;
```

```
      fclose(NAV);
      exit(0);
     }
    else fclose(sfp);
}
```

Κώδικας γραφικής αναπαράστασης

```
unit MAP_Viewer;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Menus, ExtCtrls, StdCtrls, Buttons, Printers, jpeg,Clipbrd;

type
  Node = Record
         UD:Real;
         LD:Real;
         RD:Real;
         DD:Real;
         //------
         UF:String[50];
         LF:String[50];
         RF:String[50];
         DF:String[50];
         //------
         Data:Integer;
        End;

  TForm1 = class(TForm)
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    Open1: TMenuItem;
    N1: TMenuItem;
    Exit1: TMenuItem;
```

```
    Printer1: TMenuItem;
    Setup1: TMenuItem;
    About1: TMenuItem;
    Panel1: TPanel;
    OpenDialog1: TOpenDialog;
    PrinterSetupDialog1: TPrinterSetupDialog;
    PrintDialog1: TPrintDialog;
    N2: TMenuItem;
    PrintMap1: TMenuItem;
    Image1: TImage;
    BitBtn3: TBitBtn;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    procedure Exit1Click(Sender: TObject);
    procedure Setup1Click(Sender: TObject);
    procedure About1Click(Sender: TObject);
    procedure Open1Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
    procedure PrintMap1Click(Sender: TObject);
    procedure Image1MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
    N:Integer;
    MAP:Array [1..100,1..100] of Node;
    procedure ShowMap;
  end;

var
  Form1: TForm1;
```

```
implementation

uses Unit2;

{$R *.DFM}

procedure TForm1.Exit1Click(Sender: TObject);
begin
 Halt(0);
end;

procedure TForm1.Setup1Click(Sender: TObject);
begin
 PrinterSetupDialog1.Execute;
end;

procedure TForm1.About1Click(Sender: TObject);
begin
 MessageDlg('Map Viewer by Michalis Diamantakis 2002.', mtInformation,
     [mbOk], 0);
end;

procedure TForm1.Open1Click(Sender: TObject);
Var F:TextFile;
   I,J:Integer;
begin
 If OpenDialog1.Execute then
  Begin
   AssignFile(F,OpenDialog1.FileName);
   Reset(F);
   Readln(F,N);
   For I:=1 to N do
    For J:=1 to N do
```

```
    Begin
     Read(F,MAP[I,J].Data);
     Readln(F,MAP[I,J].UD,MAP[I,J].LD,MAP[I,J].RD,MAP[I,J].DD);
     Readln(F,MAP[I,J].UF);
     Readln(F,MAP[I,J].LF);
     Readln(F,MAP[I,J].RF);
     Readln(F,MAP[I,J].DF);
    End;
  CloseFile(F);
  //-------------
  ShowMap;
 End;
end;

procedure TForm1.BitBtn1Click(Sender: TObject);
Var F:TextFile;
   I,J:Integer;
begin
 If OpenDialog1.Execute then
  Begin
   AssignFile(F,OpenDialog1.FileName);
   Reset(F);
   Readln(F,N);
   For I:=1 to N do
    For J:=1 to N do
     Begin
      Read(F,MAP[I,J].Data);
      Readln(F,MAP[I,J].UD,MAP[I,J].LD,MAP[I,J].RD,MAP[I,J].DD);
      Readln(F,MAP[I,J].UF);
      Readln(F,MAP[I,J].LF);
      Readln(F,MAP[I,J].RF);
      Readln(F,MAP[I,J].DF);
     End;
   CloseFile(F);
```

```
   //-------------
    ShowMap;
  End;
end;

procedure TForm1.ShowMap;
Var I,J,X1,X2,Y1,Y2:Integer;
    TH,TW:Integer;
    S:String;
    TS:TSize;
begin
 Image1.Canvas.Brush.Color:=clWhite;
 Image1.Canvas.FillRect(Rect(0,0,5000,5000));
 //----------
 TH:=Round((Image1.ClientHeight-100)/N);
 TW:=Round((Image1.ClientWidth-100)/N);
 Image1.Canvas.Rectangle(50,50,Image1.ClientWidth-50,Image1.ClientHeight-50);
 //----------
 Image1.Canvas.MoveTo(50,20);
 Image1.Canvas.LineTo(50,40);
 Image1.Canvas.MoveTo(Image1.ClientWidth-50,20);
 Image1.Canvas.LineTo(Image1.ClientWidth-50,40);
 Image1.Canvas.MoveTo(50,30);
 Image1.Canvas.LineTo(Image1.ClientWidth-50,30);
 Str(0.9*N:0:1,S);
 S:=' '+S+'m ';
 TS:=Image1.Canvas.TextExtent(S);
 Image1.Canvas.TextOut(Round((Image1.ClientWidth-TS.cx)/2),30-
Round(TS.cy/2),S);
 Image1.Canvas.MoveTo(20,50);
 Image1.Canvas.LineTo(40,50);
 Image1.Canvas.MoveTo(20,Image1.ClientHeight-50);
 Image1.Canvas.LineTo(40,Image1.ClientHeight-50);
 Image1.Canvas.MoveTo(30,50);
```

```
Image1.Canvas.LineTo(30,Image1.ClientHeight-50);
Str(0.9*N:0:1,S);
S:=S+'m';
Image1.Canvas.TextOut(30-Round(TS.cx/2),Round((Image1.ClientHeight-
TS.cy)/2),S);
//----------
Image1.Canvas.Pen.Style:=psDot;
For I:=1 to N-1 do
 Begin
  Image1.Canvas.MoveTo(50,50+TH*I);
  Image1.Canvas.LineTo(Image1.ClientWidth-50,50+TH*I);
  Image1.Canvas.MoveTo(50+TW*I,50);
  Image1.Canvas.LineTo(50+TW*I,Image1.ClientHeight-50);
 End;
Image1.Canvas.Pen.Style:=psSolid;
//----------
For I:=1 to N do
For J:=1 to N do
If (MAP[I,J].Data=1) then
 Begin
  X1:=51+TW*(J-1);
  X2:=X1+TW;
  Y1:=51+TH*(I-1);
  Y2:=Y1+TH;
  //---------
  If (MAP[I,J].UD<0.9) then Y1:=Y1+Round(TH*MAP[I,J].UD/0.9);
  If (MAP[I,J].DD<0.9) then Y2:=Y2-Round(TH*MAP[I,J].DD/0.9);
  If (MAP[I,J].LD<0.9) then X1:=X1+Round(TW*MAP[I,J].LD/0.9);
  If (MAP[I,J].RD<0.9) then X2:=X2-Round(TW*MAP[I,J].RD/0.9);
  //---------
  Image1.Canvas.Brush.Color:=clRed;
  Image1.Canvas.FillRect(Rect(X1,Y1,X2,Y2));
 End;
//----------
```

```
end;

procedure TForm1.BitBtn2Click(Sender: TObject);
begin
 ShowMap;
end;

procedure TForm1.BitBtn3Click(Sender: TObject);
Var R:TRect;
begin
 If PrintDialog1.Execute then
  Begin
   R:=Rect(0,0,Image1.ClientWidth,Image1.ClientHeight);
   Printer.BeginDoc;
   Printer.Canvas.CopyRect(R,Image1.Canvas,R);
   Printer.EndDoc;
  End;
end;

procedure TForm1.PrintMap1Click(Sender: TObject);
Var R:TRect;
begin
 If PrintDialog1.Execute then
  Begin
   R:=Rect(0,0,Image1.ClientWidth,Image1.ClientHeight);
   Printer.BeginDoc;
   Printer.Canvas.CopyRect(R,Image1.Canvas,R);
   Printer.EndDoc;
  End;
end;

procedure TForm1.Image1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
```

```
Var TH,TW,I,J:Integer;
    JPEG1:TJPEGImage;
    MyFormat : Word;
    AData1,APalette1 : Thandle;
    AData2,APalette2 : Thandle;
    AData3,APalette3 : Thandle;
    AData4,APalette4 : Thandle;
begin
 TH:=Round((Image1.ClientHeight-100)/N);
 TW:=Round((Image1.ClientWidth-100)/N);
 //--------
 I:=Round((Y-50)/TH)+1;
 J:=Round((X-50)/TW)+1;
 //--------
 Form2.StaticText5.Visible:=True;
 Form2.StaticText6.Visible:=True;
 Form2.StaticText7.Visible:=True;
 Form2.StaticText8.Visible:=True;

 Form2.Image1.Visible:=False;
 Form2.Image2.Visible:=False;
 Form2.Image3.Visible:=False;
 Form2.Image4.Visible:=False;
 //--------
 If (I>=1)and(I<=N)and(J>=1)and(J<=N)and(MAP[I,J].Data=1) then
  Begin
   If (MAP[I,J].UF<>'')and FileExists(MAP[I,J].UF) then
    Begin
     JPEG1 := TJPEGImage.Create;
     JPEG1.LoadFromFile(MAP[I,J].UF);
     JPEG1.SaveToClipBoardFormat(MyFormat,AData1,APalette1);
     ClipBoard.SetAsHandle(MyFormat,AData1);
     Form2.Image1.Picture.LoadFromClipboardFormat(MyFormat,AData1,APalette1);
     Form2.StaticText5.Visible:=False;
```

```
 Form2.Image1.Visible:=True;
 JPEG1.Free;
End;
If (MAP[I,J].DF<>'')and FileExists(MAP[I,J].DF) then
 Begin
 JPEG1 := TJPEGImage.Create;
 JPEG1.LoadFromFile(MAP[I,J].DF);
 JPEG1.SaveToClipBoardFormat(MyFormat,AData2,APalette2);
 ClipBoard.SetAsHandle(MyFormat,AData2);
 Form2.Image2.Picture.LoadFromClipboardFormat(MyFormat,AData2,APalette2);
 Form2.StaticText6.Visible:=False;
 Form2.Image2.Visible:=True;
 JPEG1.Free;
End;
If (MAP[I,J].LF<>'')and FileExists(MAP[I,J].LF) then
 Begin
 JPEG1 := TJPEGImage.Create;
 JPEG1.LoadFromFile(MAP[I,J].LF);
 JPEG1.SaveToClipBoardFormat(MyFormat,AData3,APalette3);
 ClipBoard.SetAsHandle(MyFormat,AData3);
 Form2.Image3.Picture.LoadFromClipboardFormat(MyFormat,AData3,APalette3);
 Form2.StaticText7.Visible:=False;
 Form2.Image3.Visible:=True;
 JPEG1.Free;
End;
If (MAP[I,J].RF<>'')and FileExists(MAP[I,J].RF) then
 Begin
 JPEG1 := TJPEGImage.Create;
 JPEG1.LoadFromFile(MAP[I,J].RF);
 JPEG1.SaveToClipBoardFormat(MyFormat,AData4,APalette4);
 ClipBoard.SetAsHandle(MyFormat,AData4);
 Form2.Image4.Picture.LoadFromClipboardFormat(MyFormat,AData4,APalette4);
 Form2.StaticText8.Visible:=False;
 Form2.Image4.Visible:=True;
```

```
     JPEG1.Free;
    End;
   //-----------
   Form2.ShowModal;
  End;
end;

end.
```

Βιβλιογραφία

[1] Aekaterinidis J., K. Kostoulakis, L. Doitsidis, K. P. Valavanis, N. C. Tsourveloudis, 'An Interface System for Real-Time Mobile Robot Environment Mapping using Sonar Sensors', *WSEAS Transactions on Systems,* vol. 4, no. 2, pp. 927-933, 2003.

[2] Tsourveloudis N. C., K. P. Valavanis, T. Hebert, 'Autonomous Vehicle Navigation Utilizing Electrostatic Potential Fields and Fuzzy Logic', *IEEE Transactions on Robotics and Automation,* vol. 17, no 4, pp. 490-497, 2001.

[3] Tom Swan's.Borland C++ 3.1 SAMS PREMIER.

[4] Tom Swan's.Borland C++ 5 premier edition SAMS PREMIER.

[5] mobility