ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΡΟΜΠΟΤΙΚΗ ΙΙ : ΕΥΦΥΗ ΡΟΜΠΟΤΙΚΑ ΣΥΣΤΗΜΑΤΑ ΚΑΙ ΚΥΤΤΑΡΑ

ΕΞΑΜΗΝΙΑΙΑ ΕΡΓΑΣΙΑ- ΜΕΡΟΣ 1^{Ο}

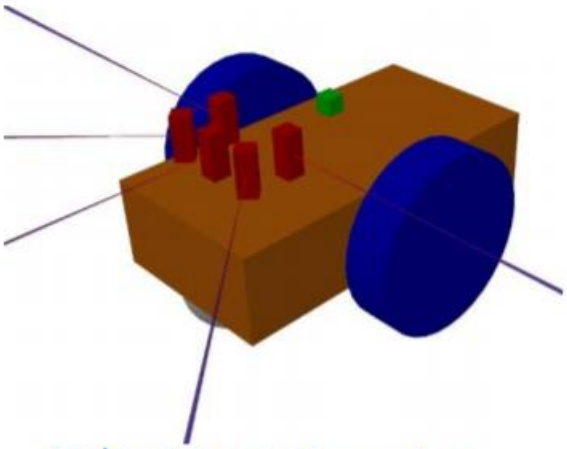
ΘΕΜΑ: ΠΑΡΑΚΟΛΟΥΘΗΣΗ ΕΜΠΟΔΙΟΥ (WALL FOLLOWING) ΣΕ ΑΓΝΩΣΤΟ ΠΕΡΙΒΑΛΛΟΝ ΜΕ ΧΡΗΣΗ ΑΙΣΘΗΤΗΡΩΝ ΑΠΟΣΤΑΣΗΣ ΥΠΕΡΗΧΩΝ.

ΑΛΑΜΑΝΟΣ ΙΩΑΝΝΗΣ (03115047)

ΔΙΑΜΑΝΤΗ ΙΩΑΝΝΑ (03115035)

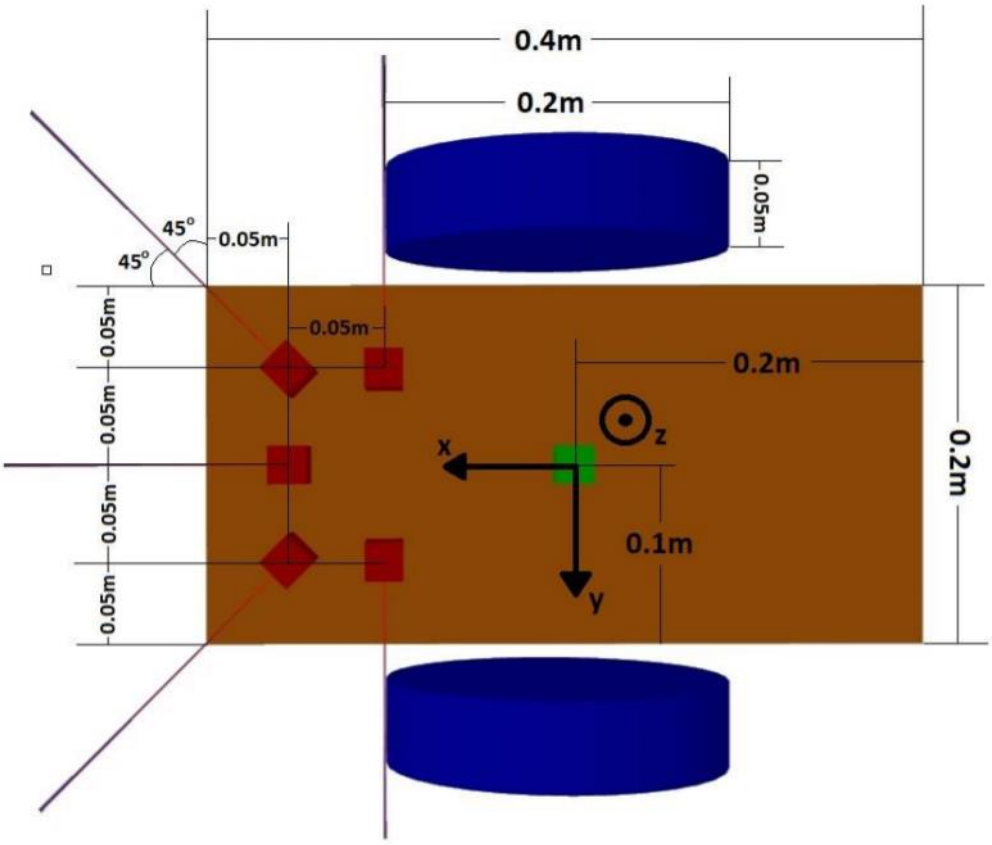
Εισαγωγή

Σκοπός της εργασίας είναι η υλοποίηση ενός αλγορίθμου για την παρακολούθηση-αποφυγή εμποδίων ενός ρομπότ κατά την εκτέλεση μίας πλήρους περιφοράς, κινούμενο παράλληλα στα εμπόδια αυτά (wall following) ,διατηρώντας σταθερή απόσταση, σε συνθήκες άγνωστου περιβάλλοντος. Η ρομποτική διάταξη που θα χρησιμοποιηθεί κατά την υλοποίηση της εργασίας, καθώς και τα χαρακτηριστικά της φαίνονται στα παρακάτω σχήματα:



Εικόνα 1 Ρομπότ Προσομοίωσης

Πλαίσια αναφοράς και διαστάσεις



Εικόνα 2 Διαστάσεις και πλαίσια αναφοράς της ρομποτικής διάταξης προσομοίωσης

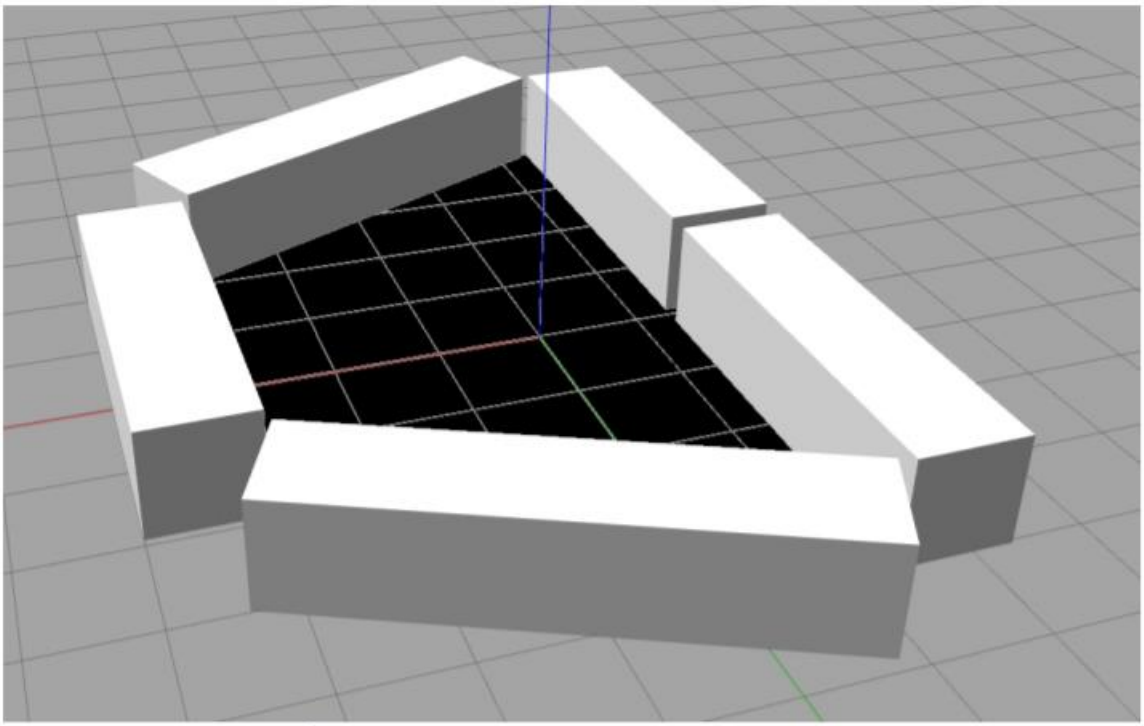
Πρόκειται για ένα ρομπότ διαφορικής οδήγησης (differential drive) με δύο τροχούς διαμέτρου 20cm. Η συγκεκριμένη ρομποτική διάταξη με διαστάσεις όπως φαίνονται στην εικόνα 2, σχεδιάστηκε σε περιβάλλον προσομοίωσης και εξοπλίστηκε με καταλλήλως για τους σκοπούς της εργασίας με τους ακόλουθους αισθητήρες:

- 5 αισθητήρες υπερήχων σόναρ, οι οποίοι μετρούν απόσταση από εμπόδια
- Ένα IMU (Inertial Measurement Unit) 6 βαθμών ελευθερίας (dof), το οποίο μετράει γραμμικές και στροφικές επιταχύνσεις καθώς κα περιστροφή γύρω από κάθε άξονα.

Η εκτέλεση των προγραμμάτων για τον έλεγχο του ρομπότ γίνεται σε περιβάλλον ROS (Robot Operating System), το οποίο υποστηρίζεται από λειτουργικά συστήματα τύπου Unix (εμείς το υλοποιήσαμε πάνω σε Linux Ubuntu 16.04). Το πλαίσιο εργασίας ROS είναι μια συλλογή από εργαλεία, βιβλιοθήκες και συμβάσεις εργασίας, η οποία έχει ως βασικό στόχο να απλοποιήσει την δημιουργία σύνθετου και αξιόπιστου ρομποτικού λογισμικού.

Η προγραμματιστική υλοποίηση για τις ανάγκες της παρούσας εργασίας έγινε με την χρήση του περιβάλλοντος προσομοίωσης Gazebo. Το συγκεκριμένο περιβάλλον προσομοίωσης παρέχει ρεαλιστικές υλοποιήσεις ενός μεγάλου αριθμού ρομποτικών διατάξεων και μπορεί ως εκ τούτου να χρησιμοποιηθεί για την ανάπτυξη και δοκιμή των προγραμμάτων ελέγχου του ρομπότ και την ολοκλήρωση των στόχων της εργασίας που περιγράφηκαν παραπάνω. Ο συγκεκριμένος τύπος προσομοιωτή έχει τη δυνατότητα συνεργασίας με το ROS.

Το περιβάλλον στο οποίο καλείται το ρομπότ να εκτελέσει μία πλήρη περιστροφή φαίνεται στην εικόνα 3.



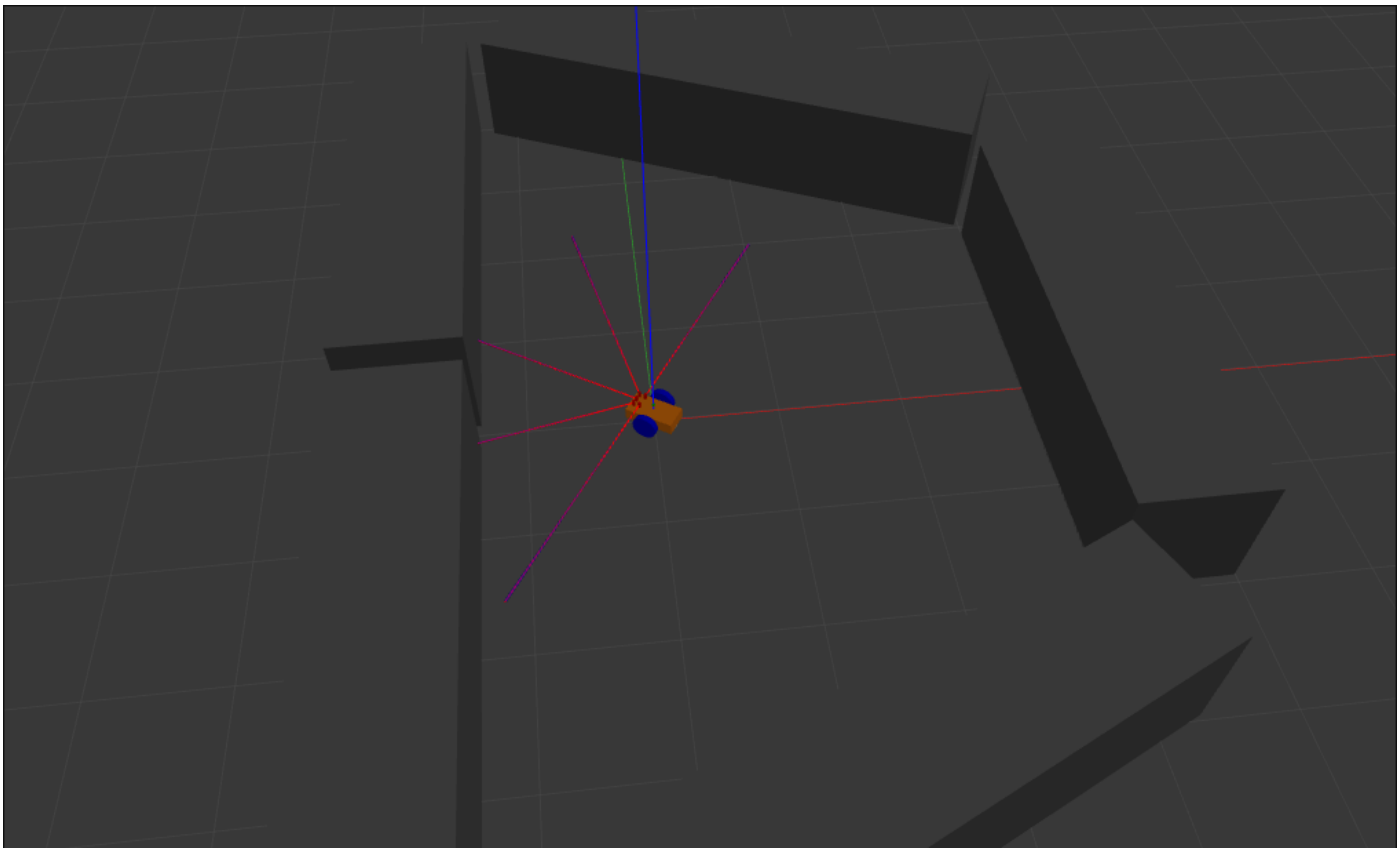
Εικόνα 3 Διάταξη εμποδίων στο περιβάλλον Gazebo

Η επιθυμητή απόσταση του τοίχου από το ρομπότ επιλέξαμε να είναι 0.3m από το L sonar, δηλαδή 0.2m από τον αριστερό τροχό και 0.35m από το κέντρο μάζας.

Τέλος, με βάση τα δύο τελευταία ψηφία των αριθμών μητρώου X1, X2 θέτουμε τον αρχικό προσανατολισμό του ρομπότ στον χώρο με περιστροφή ως προς τον Z άξονα ως εξής:

$$angle = mod(X1+X2,\pi)\ (rad)$$

Επίσης για ζυγό άθροισμα των X1, X2 προκύπτει ότι το ρομπότ θα εκτελέσει περιστροφή με ωρολογιακή φορά, ενώ σε αντίθετη περίπτωση με αντι-ωρολογιακή. Έτσι για X1 = 7, X2 = 5 προκύπτει ωρολογιακής φοράς περιστροφή με αρχική γωνία προσανατολισμού ίση με θ = 2.5752 rad. Ο αρχικός προσανατολισμός του ρομπότ φαίνεται παρακάτω:



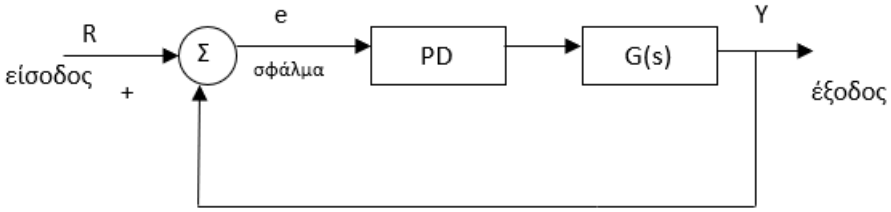
Η υλοποίηση του αλγορίθμου ελέγχου του ρομπότ έγινε με δύο διαφορετικούς τρόπους, οι οποίοι διαφέρουν στον τρόπο με τον οποίο ελέγχεται ο προσανατολισμός του και θα εξηγηθούν αναλυτικά παρακάτω.

1η Μέθοδος

Θεωρητική ανάλυση του PD ελεγκτή

Ο πρώτος τρόπος υλοποίησης ελέγχου του ρομπότ, βασίζεται στον σχεδιασμό ενός PD ελεγκτή για την ρύθμιση του προσανατολισμού του ρομπότ σε σχέση με τα εμπόδια και την παράλληλη με τον τοίχο κίνησή του. Ακολουθεί μία σύντομη περιγραφή ενός PD ελεγκτή:

Έστω ότι έχουμε ένα σύστημα με ανάδραση όπως το παρακάτω:



Τότε το σήμα (u) αμέσως μετά τον PD ελεγκτή (controller) θα έχει τη μορφή:

u=Kp (x_d-x_r)+ Kd ($\frac{d(x_d)}{dt} - \frac{d(x_r)}{dt}$), όπου το x_d εκφράζει την πραγματική θέση, το x_r την επιθυμητή θέση, το $\frac{d(x_d)}{dt}$την πραγματική ταχύτητα και το $\frac{d(x_r)}{dt}$ την επιθυμητή ταχύτητα. Ισχύει x_d-x_r=e και $\frac{d(x_d)}{dt}-\frac{d(x_r)}{dt}=\frac{d(e)}{dt}$ όπου e το σφάλμα.

Η χρήση ενός αναλογικού ελεγκτή (Kp), θα έχει ως αποτέλεσμα την ελάττωση του χρόνου ανύψωσης και την μείωση, αλλά ποτέ την εξάλειψη, του μόνιμου σφάλματος. Επομένως ο αναλογικός ελεγκτής (Kp) επηρεάζει τόσο τη μεταβατική όσο και τη μόνιμη κατάσταση του συστήματος. Ο διαφορικός έλεγχος (Kd) θα έχει ως αποτέλεσμα την αύξηση της σταθερότητας του συστήματος, μειώνοντας την υπερύψωση και βελτιώνοντας τη μεταβατική απόκριση. Στην ουσία ο διαφορικός έλεγχος (Kd) «φρενάρει» το σύστημά μας και επηρεάζει μόνο τη μεταβατική κατάσταση του συστήματος.

Αλγόριθμος υλοποίησης

1) Ανάλυση των φάσεων

Ο πρώτος τρόπος υλοποίησης του ελέγχου, βασίζεται στη λειτουργία ενός PD ελεγκτή. Στις περιπτώσεις όπου η χρήση του ελεγκτή δεν μας δίνει την επιθυμητή κίνηση του ρομπότ, χρησιμοποιήσαμε διαφορετικούς αλγορίθμους για την υλοποίηση, τους οποίους χωρίσαμε ανά φάση ανάλογα με την κατάσταση στην οποία βρίσκεται το έντροχο ρομπότ. Συγκεκριμένα υπάρχει η πρώτη φάση, κατά την οποία το ρομπότ βρίσκεται στην αρχική του διάταξη και θέση όπως αυτή φαίνεται παραπάνω και κινείται προς τον τοίχο για να ξεκινήσει το wall following. Στην ουσία αυτό που κάναμε ήταν να ορίσουμε πειραματικά μία χρονική στιγμή κατά την οποία το ρομπότ έχει έρθει σχετικά κοντά στον τοίχο και από εκείνη τη στιγμή και μετά θα μπούμε στον PD ελεγκτη και θα ξεκινήσει το wall following. Ο λόγος που δεν μπαίνουμε κατευθείαν στον ελεγκτή μας είναι επειδή ξεκινώντας από το κέντρο τα sonars βλέπουν πολύ μεγάλη απόσταση από τα εμπόδια και έτσι ο ελεγκτής δίνει πολύ μεγάλη γωνιακή ταχύτητα με αποτέλεσμα το ρομπότ να κάνει κύκλους γύρω από τον εαυτό του. Πέρα από τη φάση αυτή υπάρχει και η δεύτερη φάση κατά την οποία το ρομπότ έχει ξεκινήσει το wall following και βλέπει κάποιο εμπόδιο μπροστά του σε απόσταση μικρότερη μίας πειραματικής τιμής- τιμής ασφαλείας για σύγκρουση. Τέλος έχουμε ορίσει και μία τρίτη φάση κατά την οποία επιβάλλουμε στο ρομπότ να σταματήσει ομαλά αφότου έχει εκτελέσει μία πλήρη περιστροφή. Για να το επιτύχουμε αυτό ορίσαμε πειραματικά μία χρονική στιγμή στην οποία παρατηρήσαμε το ρομπότ έχει εκτελέσει μια περιστροφή.

Να επισημάνουμε ότι για την υλοποίηση του αλγορίθμου έχει αρχικά οριστεί μία τελική γραμμική και γωνιακή ταχύτητα (κατώφλι) ίση με 0.2 m/s και rad/s αντίστοιχα, η οποία επιλέχθηκε πειραματικά ώστε το ρομπότ να κινείται σχετικά γρήγορα χωρίς όμως να αποπροσανατολίζεται πολύ εξαιτίας του θορύβου που εισάγεται.

Στην πρώτη φάση ελέγχονται τα εξής:

- Στην πρώτη φάση το ρομπότ ξεκινώντας από την αρχική διάταξη επιταχύνει ομαλά τόσο τη γραμμική όσο και τη γωνιακή του ταχύτητα μέχρι να φτάσουν στην επιθυμητή τιμή και συνεχίζει την πορεία του μέχρι να φτάσει κοντά στον τοίχο (δηλαδή μέχρι τη χρονική στιγμή που έχουμε ορίσει πειραματικά), στη συνέχεια μπαίνει στον PD ελεγκτή και ξεκινάει το wall following. Η φάση αυτή δεν χρησιμεύει ξανά στο υπόλοιπο του αλγορίθμου. Είναι σημαντικό να τονίσουμε ότι εισάγαμε πέρα από την γραμμική ταχύτητα που είναι προφανώς απαραίτητη για να πλησιάσει στον τοίχο, και γωνιακή ταχύτητα για το λόγο ότι σε περίπτωση που βάζαμε μηδενική γωνιακή το αριστερά sonar το ρομπότ έπαιρνε “λάθος” μέτρηση καθώς μέτραγε στο σημείο που οι δύο τοίχοι έχουν κενό μεταξύ τους, και έτσι δεν είχαμε ομαλή συμπεριφορά μόλις μπαίναμε στον PD ελεγκτή. Ο τρόπος με τον οποίο σχεδιάστηκε η ομαλή επιτάχυνση αλλά και επιβράδυνση (για τη

φάση 2) φαίνεται ακριβώς παρακάτω. Η σκέψη ήταν ότι θέλουμε να έχουμε συνέχεια και στην επιτάχυνση για αυτό πήραμε πολυώνυμα τρίτου βαθμού.

Εξισώσεις ταχυτήτων:

$V = V_0 + a_0t + a_1t^2 + a_2t^3$, όπου $V_0 = 0$ m/s και $a_0 = 0$ m/s^2

$\omega = \omega_0 + \alpha\omega_0 t + a_1t^2 + a_2t^3$, όπου $\omega_0 = 0$ rad/s και $\alpha\omega_0 = 0$ rad/s^2.

Εξισώσεις επιτάχυνσης:

$a = a_0 + 2a_1t + 3a_2t^2$, όπου $a_0 = 0$ m/s^2

$\alpha\omega = \alpha\omega_0 + 2a_1t + 3a_2t^2$, όπου $\alpha\omega_0 = 0$ rad/s^2

Θεωρώντας ότι θέλουμε σε $\delta t = 0.5$ sec να πετύχουμε την επιθυμητή ταχύτητα (0.2), εφαρμόζουμε τις τελικές συνθήκες $V(0.5) = 0.2$ m/s και $a(0.5) = 0$ m/s^2 και προκύπτουν $a_1 = 2.4$ και $a_2 = -3.2$ για ομαλή επιτάχυνση και $a_1 = -2.4$, $a_2 = 3.2$ για ομαλή επιβράδυνση. Η περίοδος δειγματοληψίας που χρησιμοποιήσαμε είναι 0.1s, δηλαδή συμπίπτει με τη χρονική διάρκεια μίας εκτέλεσης του while loop (loop_rate(10) συνεπάγεται 10hz άρα 0.1s).

Αν λοιπόν έχουμε πετύχει τις επιθυμητές ταχύτητες, τότε τις κρατάμε σταθερές (η επιτάχυνση μηδενίζεται) , διαφορετικά τις μεταβάλλουμε ανάλογα, διατηρώντας την συνέχεια των τιμών.

Στην δεύτερη φάση ελέγχονται τα εξής:

- Η φάση αυτή επί της ουσίας μας εξασφαλίζει ότι δεν θα έχουμε κάποια σύγκρουση με πιθανά εμπόδια. Χρησιμεύει στις περιπτώσεις όπου ο PD ελεγκτής (κατ' επέκταση οι μετρήσεις των αριστερών sonars) δεν μπορεί από μόνος του να μας εξασφαλίσει ότι θα ακολουθήσουμε την επιθυμητή πορεία χωρίς να συγκρουστεί το ρομπότ με τον τοίχο. Συγκεκριμένα στην φάση αυτή μπαίνουμε στις δύο "απότομες" στροφές όπου ο επόμενος τοίχος σχηματίζει οξεία γωνία σε σχέση με τον τοίχο που κινείται το ρομπότ καθώς τα δύο αριστερά sonars στην περίπτωση αυτή εξακολουθούν να παίρνουν μέτρηση από τον τοίχο στον οποίο κινούταν το ρομπότ ακόμα και μετά την σύγκρουση. Στις δύο περιπτώσεις όπου ο τοίχος παράλληλα με τον οποίο πρόκειται να κινηθεί το ρομπότ σχηματίζει αμβλεία γωνία σε σχέση με αυτόν που κινείται, τα αριστερά sonars παίρνουν έγκαιρα μέτρηση από τον τοίχο στον οποίο πρόκειται να κινηθεί και το ρομπότ στρίβει έγκαιρα χωρίς να βγει από τον ελεγκτή. Η πειραματική τιμή λοιπόν (0.29) που επιλέξαμε για την υλοποίηση χρησιμεύει τόσο στο να εντοπίζει έγκαιρα το ρομπότ ένα εμπόδιο που βρίσκεται μπροστά του αλλά και να διατηρεί την κίνηση του σύμφωνα με τον PD ελεγκτή στις περιπτώσεις που δεν είναι αναγκαίο να μπούμε στη φάση 2 και να φρενάρουμε το ρομπότ χωρίς να είναι τόσο απότομη η στροφή.

  Σε αυτό το σημείο είναι πολύ σημαντικό να τονίσουμε ότι ο αλγόριθμος σχεδιάστηκε έτσι ώστε να διατηρείται η συνέχεια των γραμμικών και γωνιακών ταχυτήτων ακόμα και όταν μπαίνουμε στη φάση 2. Πιο συγκεκριμένα το ρομπότ μπαίνει στη φάση 2 αφού προηγουμένως έχει ολοκληρώσει την παράλληλη κίνηση του με έναν τοίχο. Αυτό σημαίνει ότι πριν μπούμε στη δεύτερη φάση ο ελεγκτής έχει φτάσει στη μόνιμη κατάσταση και έτσι έχουμε πρακτικά γωνιακή ταχύτητα (τρέχοντας σε ένα τρίτο terminal την εντολή rostopic echo /cmd_vel το επιβεβαιώσαμε και πρακτικά καθώς είδαμε πάρα πολύ μικρές ταλαντώσεις γύρω από το 0) οπότε δεν χάνεται η συνέχεια μηδενίζοντας την. Όσον αφορά τη γραμμική ταχύτητα την επιβραδύνουμε ομαλά (φάση 2) και στη συνέχεια την επιταχύνουμε με τον ίδιο τρόπο μόλις μπούμε ξανά στον ελεγκτή. Κάθε φορά που

μπαίνουμε στη φάση 2 δεν βγαίνουμε από αυτήν πριν το ρομπότ να έχει στρίψει και να παίρνουμε επιθυμητές μετρήσεις από τα αριστερά sonars (δηλαδή να βλέπουν τον τοίχο στον οποίο θα κινηθεί το ρομπότ). Αυτό συμβαίνει καθώς το μπροστινό sonar βλέπει την πειραματική τιμή ασφαλείας που έχουμε ορίσει, το ρομπότ ξεκινάει να στρίβει ενώ παράλληλα το μπροστινό sonar βλέπει ακόμα μικρότερες τιμές, και τέλος το μπροστινό sonar αρχίζει να βλέπει μεγαλύτερες τιμές όταν πλέον τα αριστερά sonars δίνουν αποστάσεις σχετικά με τον νέο τοίχο και πλέον είναι επιθυμητό να μπούμε και πάλι στον PD ελεγκτή. Οπότε και από αυτή την άποψη δεν υπάρχει κάποια πιθανότητα ασυνέχειας. Προφανώς όταν μπαίνουμε στον PD ελεγκτή έχουμε εκθετική συνέχεια.

Εδώ πρέπει να επισημάνουμε ότι σε πολύ μικρό αριθμό περιπτώσεων ενδέχεται να μπούμε στη φάση αυτή και λίγο πριν το τέλος της περιστροφής στο σημείο που υπάρχει το κενό ανάμεσα στους δύο τοίχους. Αυτό συμβαίνει σε περίπτωση που το ρομπότ υποστεί ένα μικρό σφάλμα προς τα δεξιά ακριβώς πριν αρχίσει να παίρνει τις λάθος μετρήσεις και σε συνάρτηση με το ότι θα στρίψει ακόμα δεξιότερα λόγω των λάθος μετρήσεων στη συνέχεια στρίβει πολύ απότομα αριστερά με αποτέλεσμα να πέφτει ακριβώς με τέτοια φορά ώστε το μπροστινό sonar να κοιτάει το κενό. Μόνο και μόνο για το λόγο αυτό βάλαμε και την πειραματική τιμή ασφαλείας και για το μπροστά δεξιά sonar, αλλιώς δεν έχει καμία χρησιμότητα στην υπόλοιπη προσομοίωση. Στην περίπτωση που γίνει αυτό θα έχουμε ένα απότομο σταμάτημα όσον αφορά τη γωνιακή ταχύτητα χωρίς αυτό βέβαια να επηρεάσει ιδιαίτερα την υλοποίηση μας. Να τονίσουμε ότι με κατάλληλες ρυθμίσεις του PD ελεγκτή δεν είδαμε να συμβαίνει αυτό το φαινόμενο τις αρκετές τελευταίες φορές που τρέξαμε την προσομοίωση αλλά σας το επισημαίνουμε σε περίπτωση που συμβεί κάτι τέτοιο. Σε περίπτωση που δεν τύχει κάτι τέτοιο το ρομπότ μένει αποκλειστικά στον ελεγκτή και διορθώνει κανονικά. Θα μπορούσαμε και να αγνοήσουμε εντελώς τις μετρήσεις αυτές βρίσκοντας πειραματικά τη χρονική στιγμή κατά την οποία το ρομπότ ξεκινάει τις λάθος μετρήσεις αλλά και αυτή στην οποία πλέον μετράνε σωστά τα sonars και να του επιβάλλουμε να συνεχίσει να κινείται ευθεία, αλλά προτιμήσαμε να δείτε πως διορθώνει ο ελεγκτής μας καθώς αυτός φανταστήκαμε πως ήταν και ο σκοπός του κενού μεταξύ των δύο τοίχων.

2) Αλγόριθμος υλοποίησης του PD ελεγκτή και εξισώσεις ελέγχου

Στην περίπτωση που δεν βρισκόμαστε ούτε στην πρώτη ούτε στην δεύτερη φάση, το ρομπότ εκτελεί ευθύγραμμη κίνηση παράλληλη με τον τοίχο σύμφωνα με τον PD ελεγκτή για την διατήρηση της κάθετης απόστασης του ρομπότ από τον τοίχο που βρίσκεται αριστερά. Ο ελεγκτής που εξηγείται παρακάτω δεν εκτελείται όταν το ρομπότ βρίσκεται σε κάποια από τις 2 παραπάνω φάσεις, παρά μόνο όταν αυτό φτάνει κοντά στον τοίχο και χωρίς να βλέπει κάποιο εμπόδιο μπροστά του σε μικρή απόσταση.

Ο συγκεκριμένος PD ελεγκτής υλοποιήθηκε με βάση την παρακάτω εικόνα:

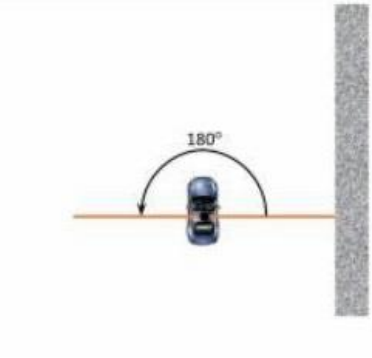


Figure 1: Lidar scan angles

Let alpha be the orientation of the car with respect to the wall. By solving the geometric problem establish alpha as tan inverse of a cos(theta) - b/asin(theta), and AB as b cos(alpha).

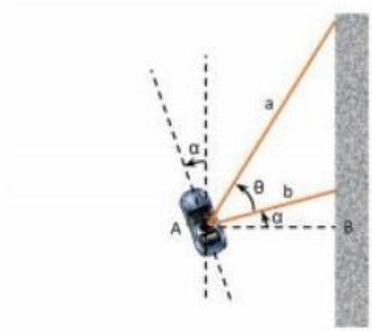


Figure 2: Calculating the orientation and distance from the wall

Συγκεκριμένα υπολογίζουμε αρχικά την γωνία α, η οποία αφορά την απόκλιση του ρομπότ από την επιθυμητή νοητή -παράλληλη στον τοίχο- ευθεία. Η γωνία αυτή υπολογίζεται εύκολα από τις ενδείξεις του αριστερού (πλευρά b του σχήματος) και του διαγώνια αριστερά (πλευρά a του σχήματος) αισθητήρων καθώς και της μεταξύ τους γωνίας θ = 45° (η οποία φαίνεται και στην εικόνα 2) ως εξής:

$$\alpha = tan^{-1}\left(\frac{acos\theta - b}{asin\theta}\right)$$

Έχοντας λοιπόν υπολογίσει την γωνία αυτή, μπορούμε να βρούμε την τωρινή κάθετη απόσταση AB του ρομπότ από τον τοίχο ως εξής: AB = bcos(α), η οποία είναι και η είσοδος του ελεγκτή. Ο ελεγκτής υπολογίζει τη διαφορά της πραγματικής από την επιθυμητή απόσταση και δίνει κατάλληλη γωνιακή ταχύτητα στο ρομπότ. Να επισημάνουμε ότι στην υλοποίηση μας τα sonars δεν βρίσκονται ακριβώς στο ίδιο σημείο όπως περιγράφεται στον αλγόριθμο (το FL sonar απέχει 0.05m από το L sonar) αλλά παρόλα αυτά ο αλγόριθμος λειτουργεί αποδοτικά αφενός γιατί η απόσταση των sonars είναι πάρα πολύ μικρή αλλά αφετέρου μας δίνει θεωρητικά και ακριβώς την επιθυμητή απόσταση καθώς όταν το ρομπότ κινείται παράλληλα με τον τοίχο το FL sonar βλέπει ακριβώς την ίδια απόσταση με αυτή που θα έβλεπε αν ήταν ακριβώς στο ίδιο σημείο με το L sonar.

Οι παράμετροι Kp , Kd του ελεγκτή υπολογίστηκαν πειραματικά ως εξής:

Ξεκινήσαμε αρχικά με μία χαμηλή τιμή Kp και Kd = 0 και αυξάνουμε σιγά σιγά την τιμή του Kp μέχρι να πάρουμε μεγάλο overshoot ή ταλαντώσεις. Στο σημείο αυτό προσθέτουμε και τον Kd όρο, τον οποίο αυξάνουμε μέχρι να πάρουμε ένα καλό αποτέλεσμα (όχι overshoot – ταλαντώσεις και σχετικά γρήγορη απόκριση). Στη συνέχεια αυξάνουμε το Kp μέχρι να μην έχουμε πλέον επιθυμητό αποτέλεσμα, όπου αυξάνουμε το Kd μέχρι να πάρουμε πάλι ένα καλό αποτέλεσμα. Συνεχίζουμε αυτή τη διαδικασία μέχρι να πετύχουμε μικρό σφάλμα στη μόνιμη κατάσταση (αν συνεχιστεί επ άπειρον θα πετύχουμε θεωρητικά μηδενικό σφάλμα μόνιμης κατάστασης).

Για την υλοποίηση του P όρου του ελεγκτή υπολογίσαμε το σφάλμα μεταξύ της επιθυμητής κάθετης απόστασης (0.3m ως προς το L sonar) και της πραγματικής κάθετης απόστασης (AB = bcos(α)). Ο D όρος του ελεγκτή υλοποιήθηκε μέσω της διαφοράς του προηγούμενου από του παρόντος σφάλματος. Έτσι λοιπόν σύμφωνα με την παραπάνω πειραματική μέθοδο ορίσαμε τις παραμέτρους kp=12 και kd=11.3 για τις οποίες παρατηρήσαμε ότι το σύστημα συμπεριφέρεται καλά τόσο ως προς το overshoot όσο και ως προς το χρόνο αποκατάστασης και το μόνιμο σφάλμα. Με τη συγκεκριμένη ρύθμιση πετύχαμε μικρό overshoot με σχετικά μικρό χρόνο αποκατάστασης και μόνιμο σφάλμα της τάξης 0.004-0.006m (σύμφωνα με τις μετρήσεις του L sonar).

Σημείωση: Το μεγαλύτερο σφάλμα που παρατηρούμε κατά την εκτέλεση της προσομοίωσης λόγω του αποπροσανατολισμού του ρομπότ είναι 0.025-0.035m και το θεωρούμε αποδεκτό.

Εξίσωση ελέγχου: velocity.angular.z = 12 * (error) + 11.3 * (error - prev_error)

Τέλος όταν το ρομπότ έχει σχεδόν εκτελέσει μία πλήρη περιφορά στο περιβάλλον προσομοίωσης (φάση 3), επιβραδύνουμε ομαλά την γραμμική ταχύτητα που έχει αναπτύξει μέχρι αυτή να γίνει μηδενική. Εφόσον όταν συμβαίνει αυτό είμαστε στην φάση του wall following, η γωνιακή ταχύτητα είναι μηδενική, όπως και παραμένει.

Στις φάσεις 1,2 και 3 όπου η κίνηση δεν ελέγχεται από τον PD ελεγκτή, σύμφωνα με τους αλγορίθμους που εξηγήσαμε παραπάνω υπολογίζουμε τις επιθυμητές ταχύτητες (ανάλογα με τη φάση που βρίσκεται το ρομπότ) και τις περνάμε κατευθείαν στις εξισώσεις ελέγχου καθώς θεωρούμε ότι οι ταχύτητες που στέλνουμε μεταφράζονται σωστά (τρέχοντας την εντολή rostopic echo /odom παρατηρήσαμε ότι υπάρχει μικρή απόκλιση σε ορισμένες χρονικές στιγμές).

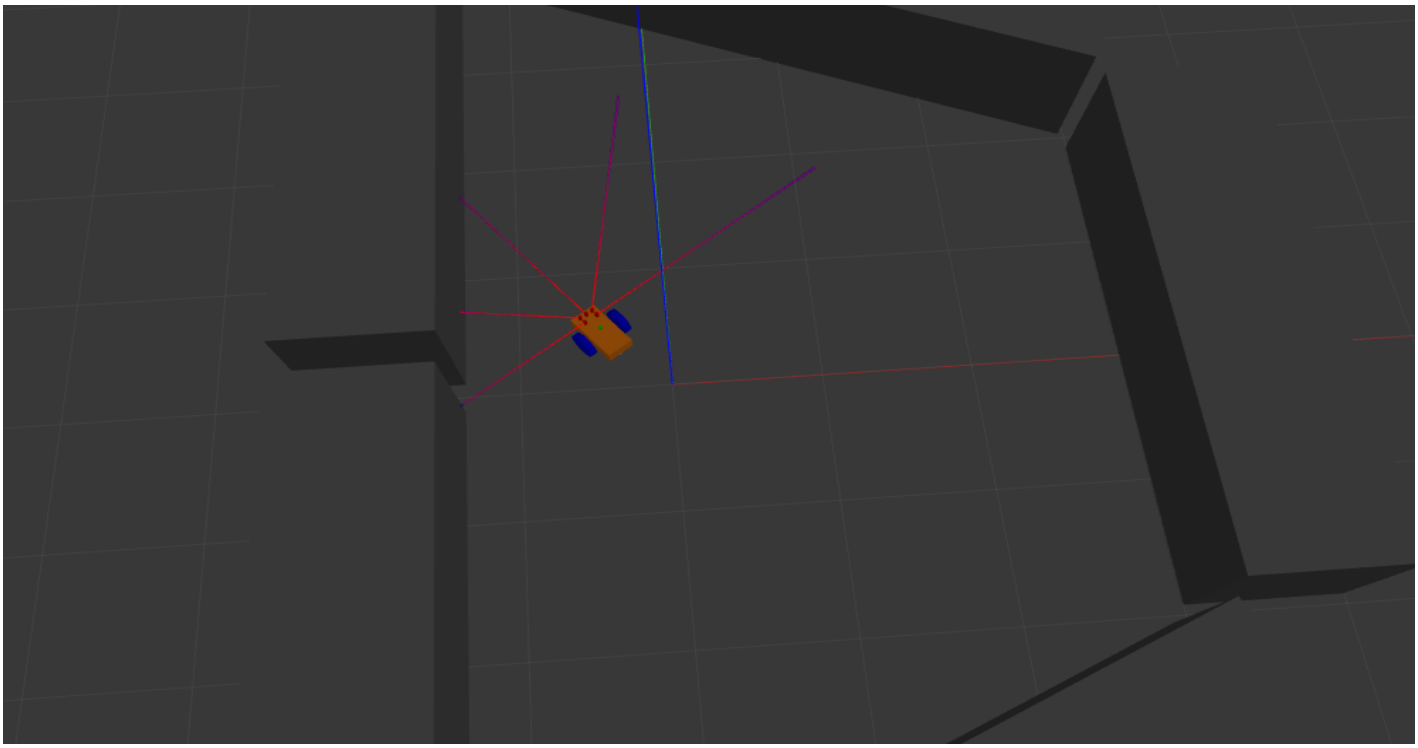
Επομένως οι εξισώσεις ελέγχου είναι: velocity.linear.x = v_d

velocity.angular.z = a_d

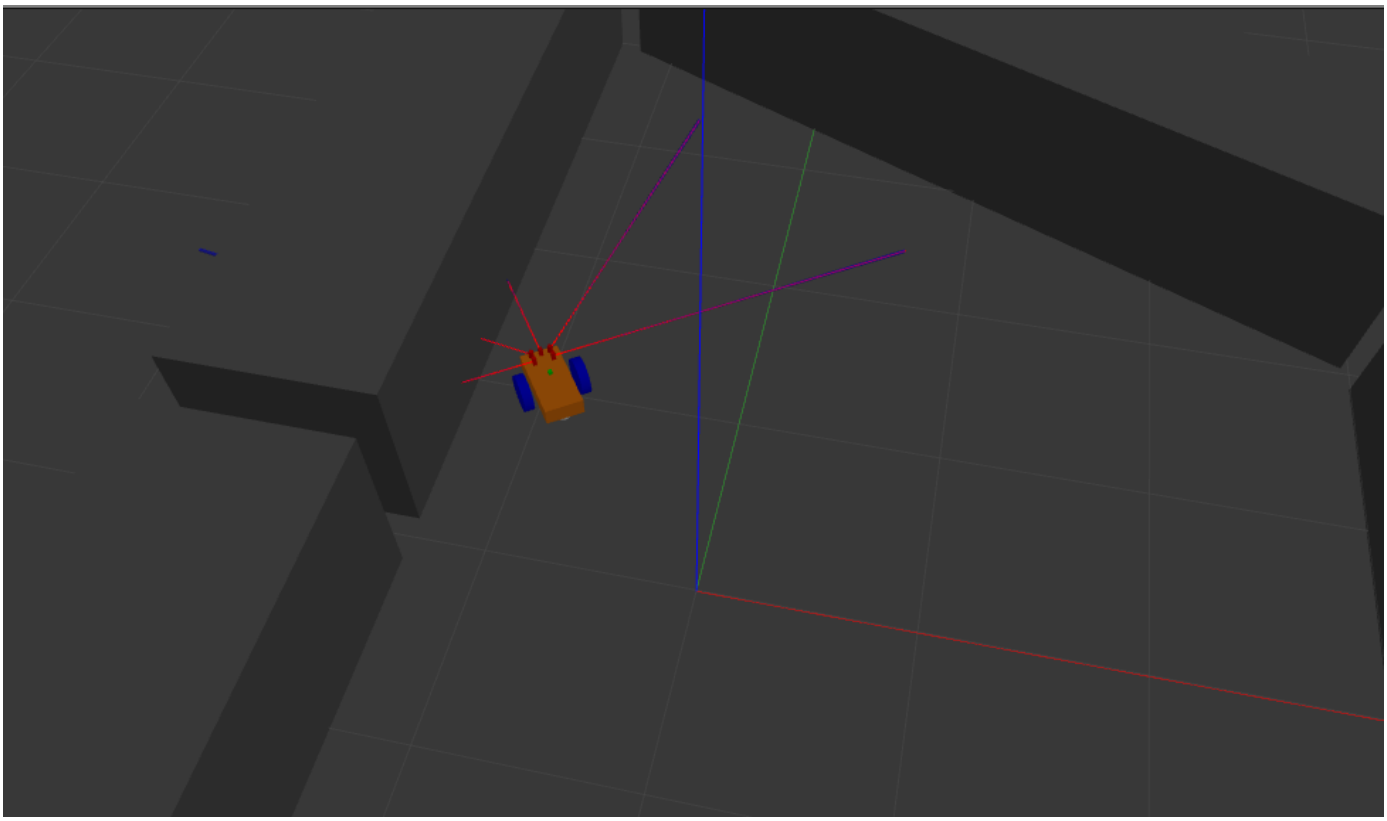
Παρακάτω βλέπουμε μερικά αντιπροσωπευτικά στιγμιότυπα της κίνησης του ρομπότ, όπου επιδεικνύεται η αποτελεσματικότητα του αλγορίθμου:

<u>Σημείωση: Η προσομοίωση διαρκεί περίπου 4 λεπτά και 35 δευτερόλεπτα (Real time).</u>

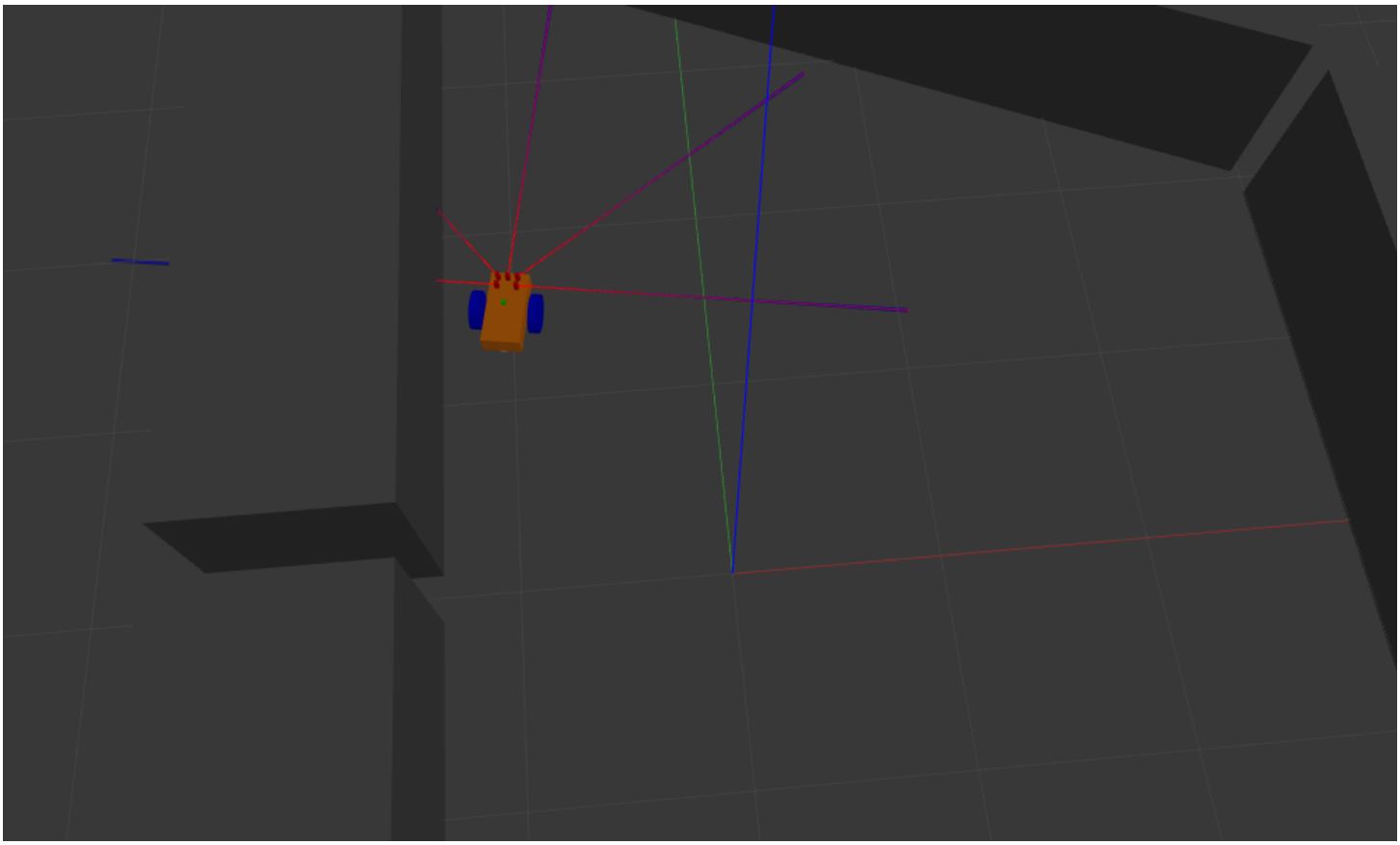
Φάση 1 -Κίνηση του ρομπότ από το κέντρο προς τον τοίχο:



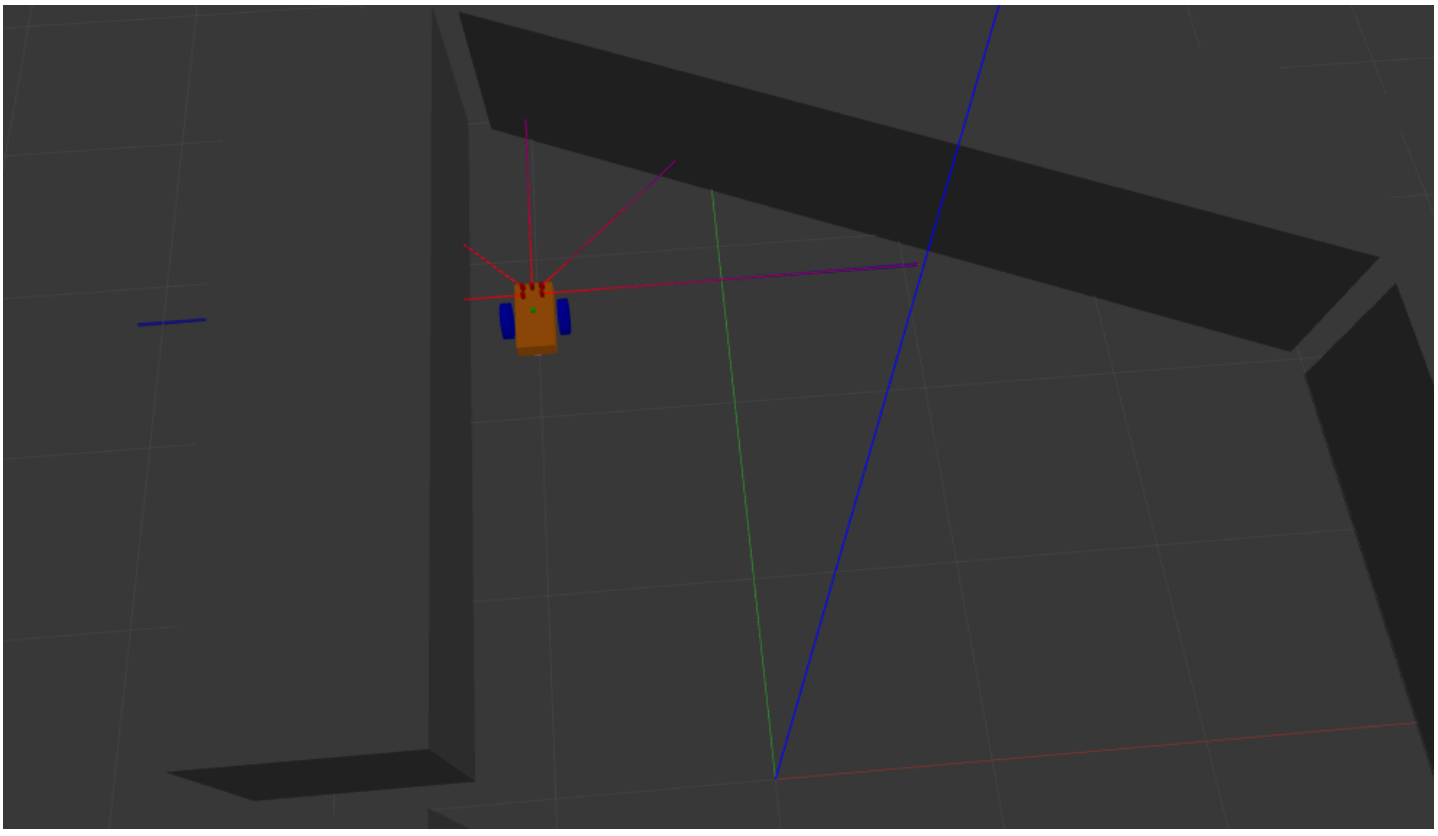
Έναρξη εκτέλεσης PD ελεγκτή:



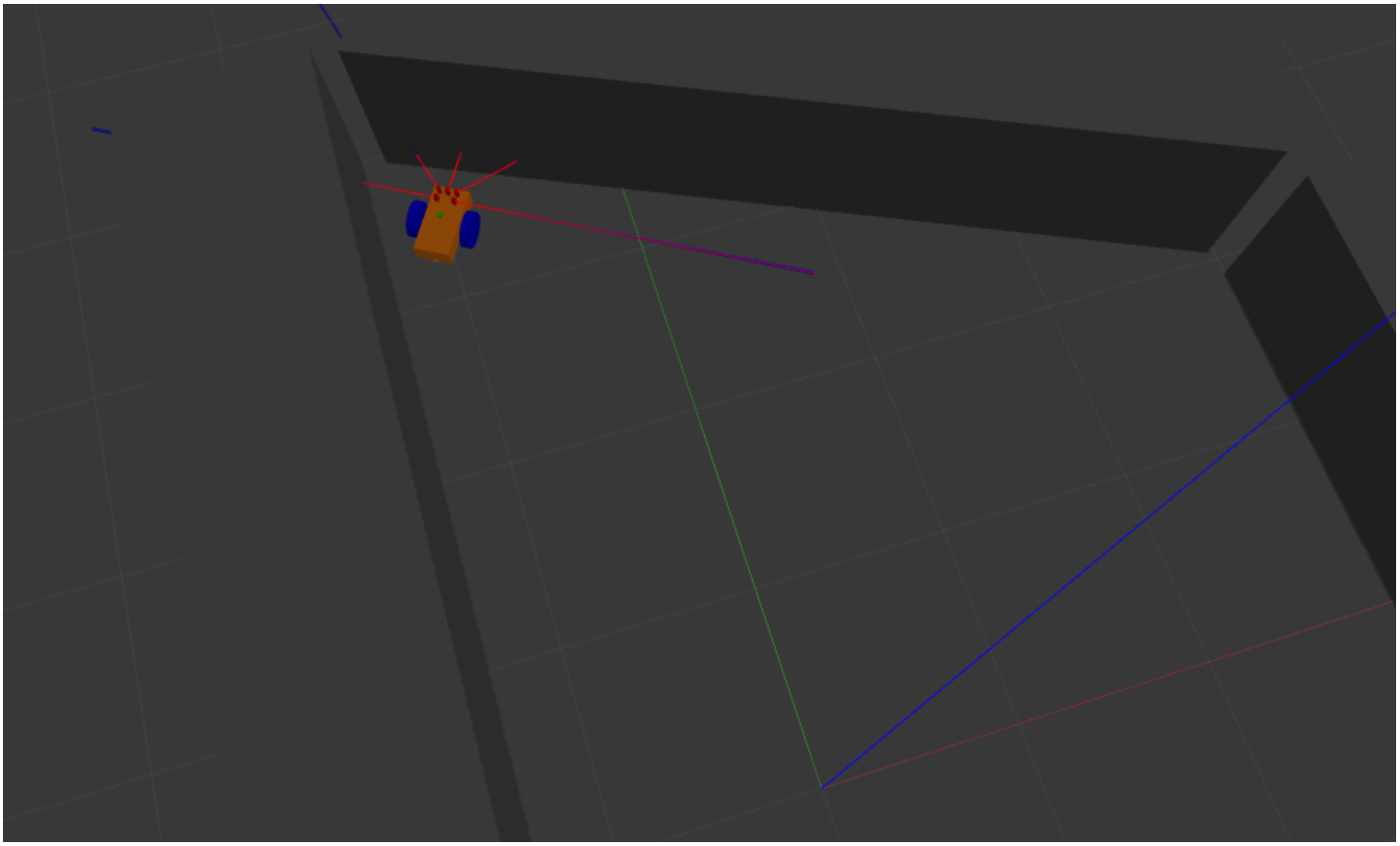
Αρχικό σφάλμα (Overshoot) ελεγκτή:



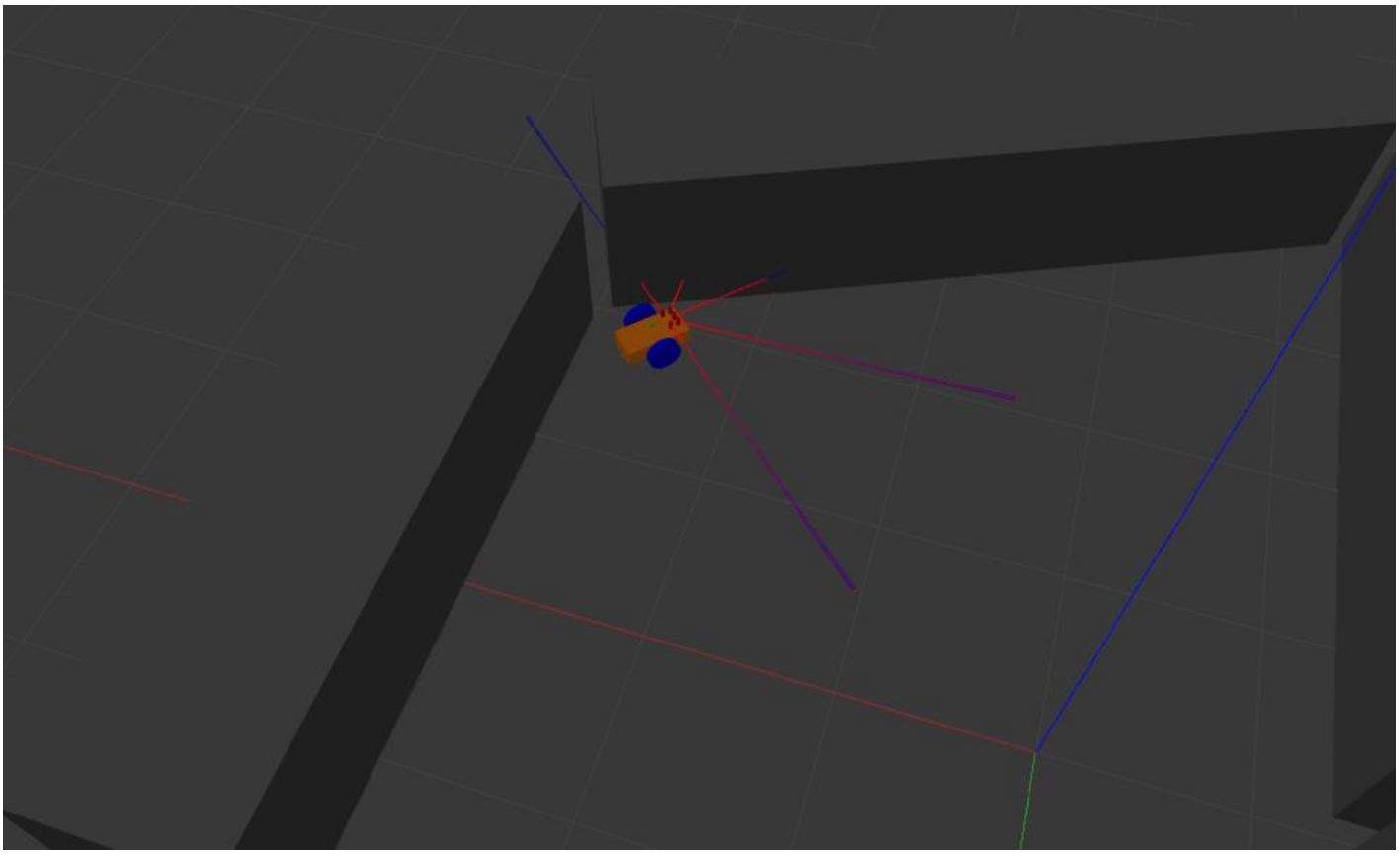
Μόνιμη κατάσταση ελεγκτή:



Φάση 2 (το ρομπότ έχει σταματήσει και στρίβει) :

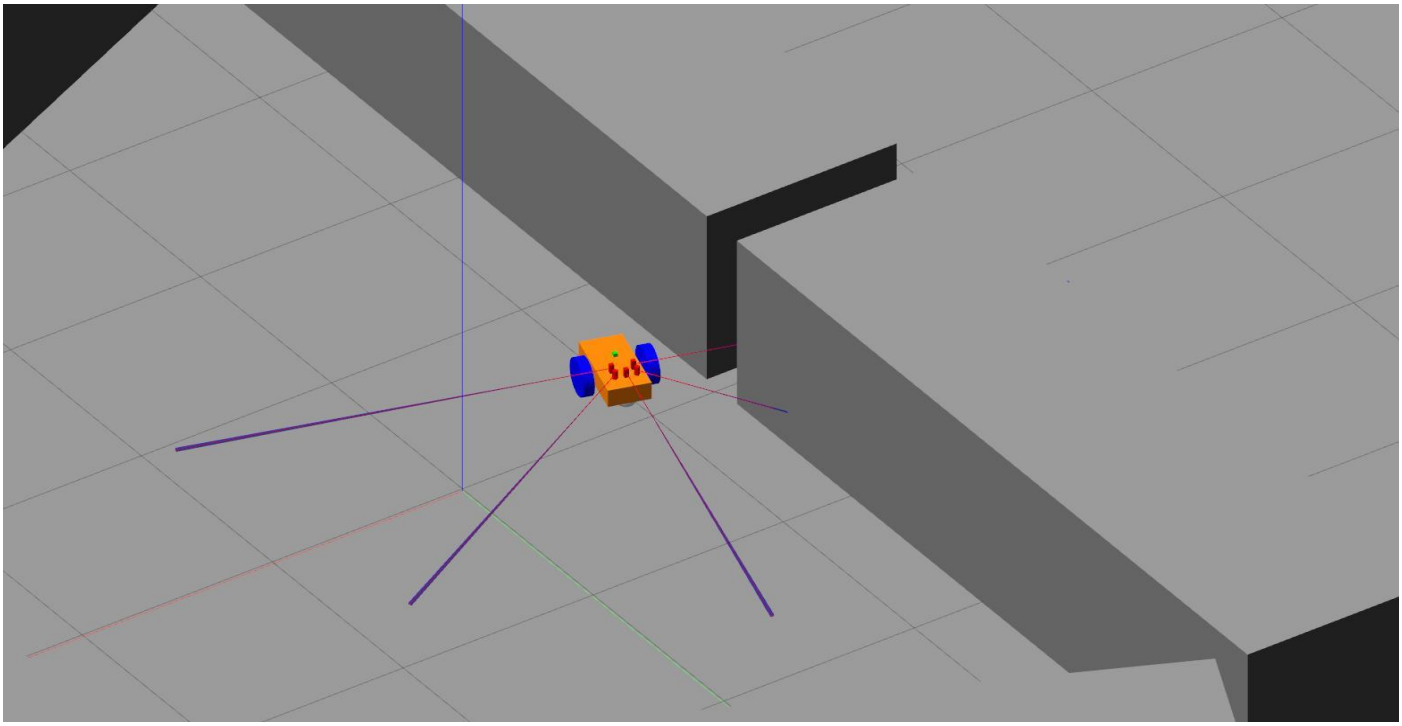


Στροφή μέσω PD ελεγκτή (έχουμε και γραμμική και γωνιακή ταχύτητα):

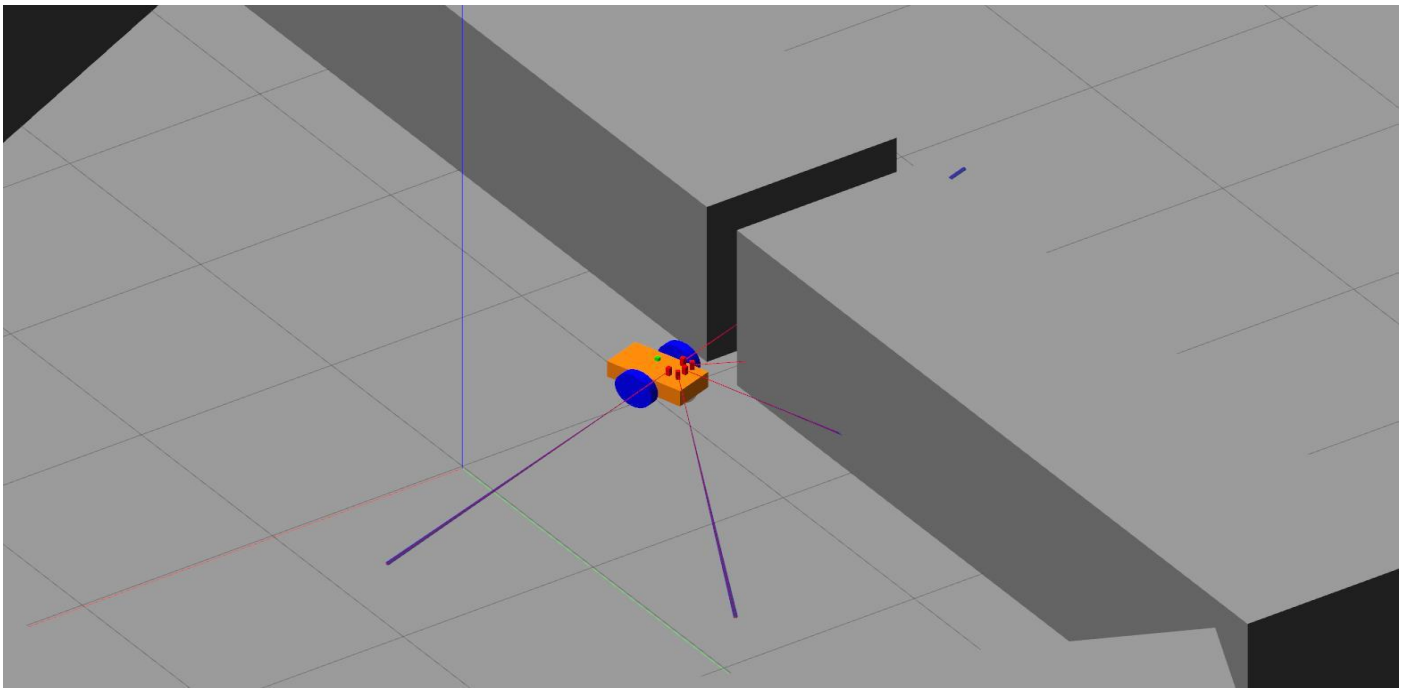


Συμπεριφορά του ρομπότ στο σημείο που παίρνει λάθος μετρήσεις:

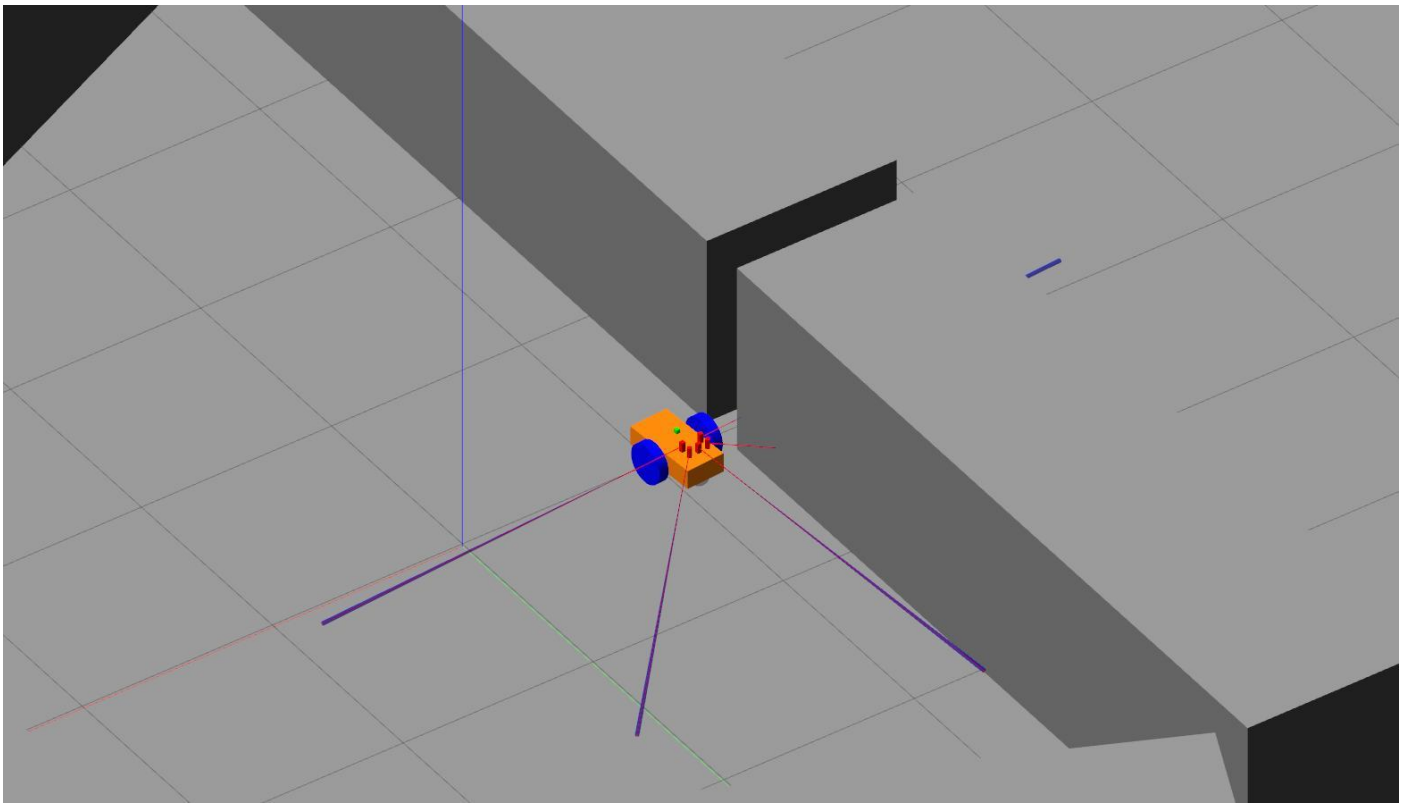
Αποπροσανατολισμός από λάθος μέτρηση του L sonar

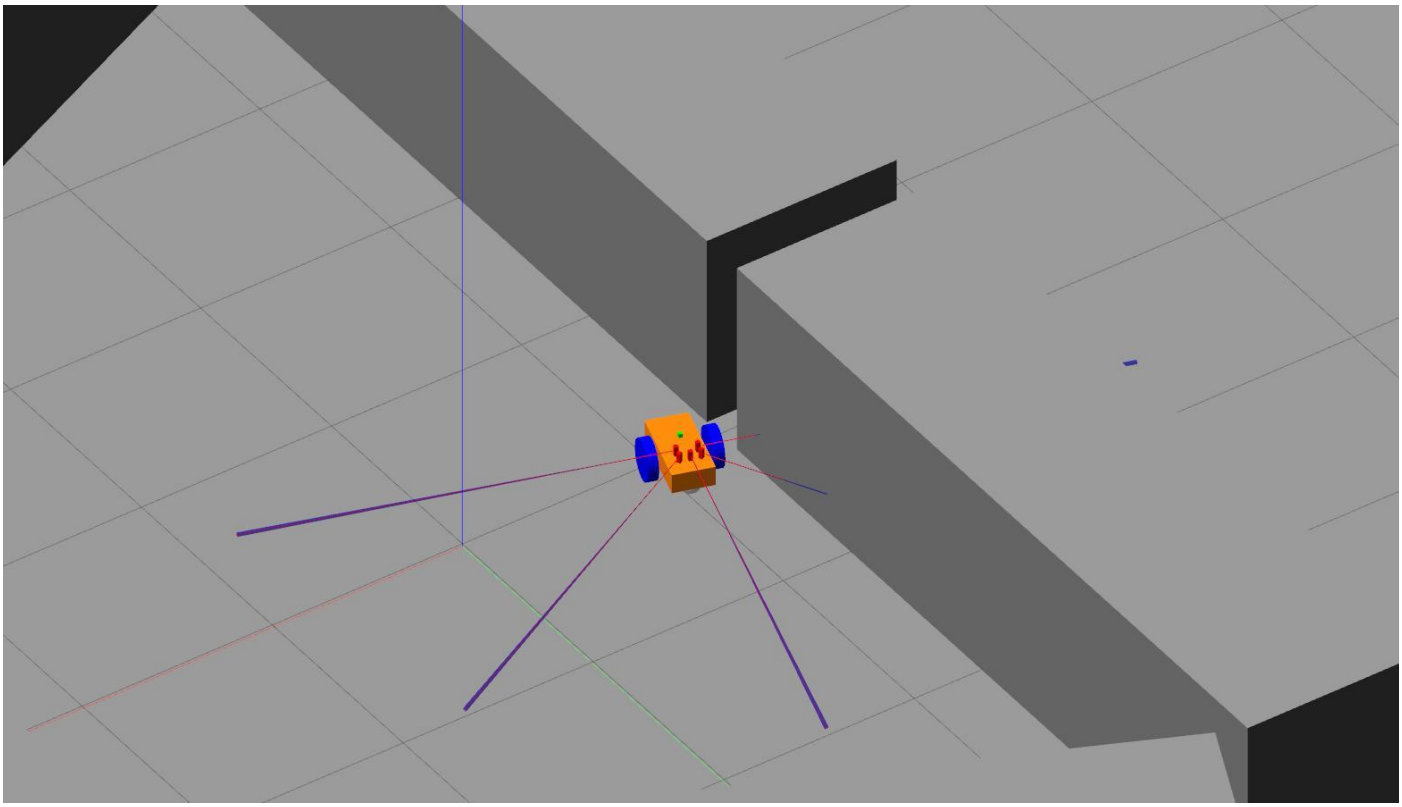


Ξεκινάει η διόρθωση

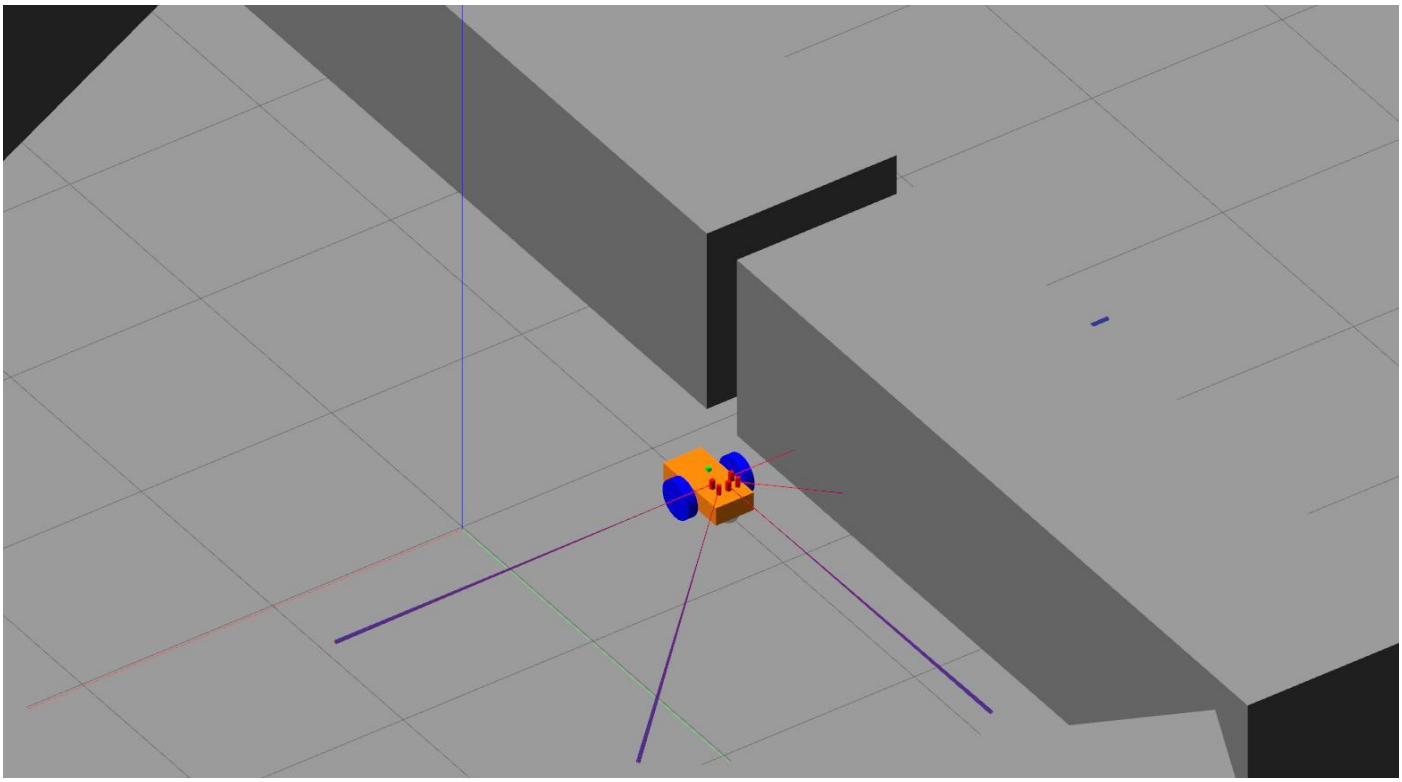


Διορθώνει

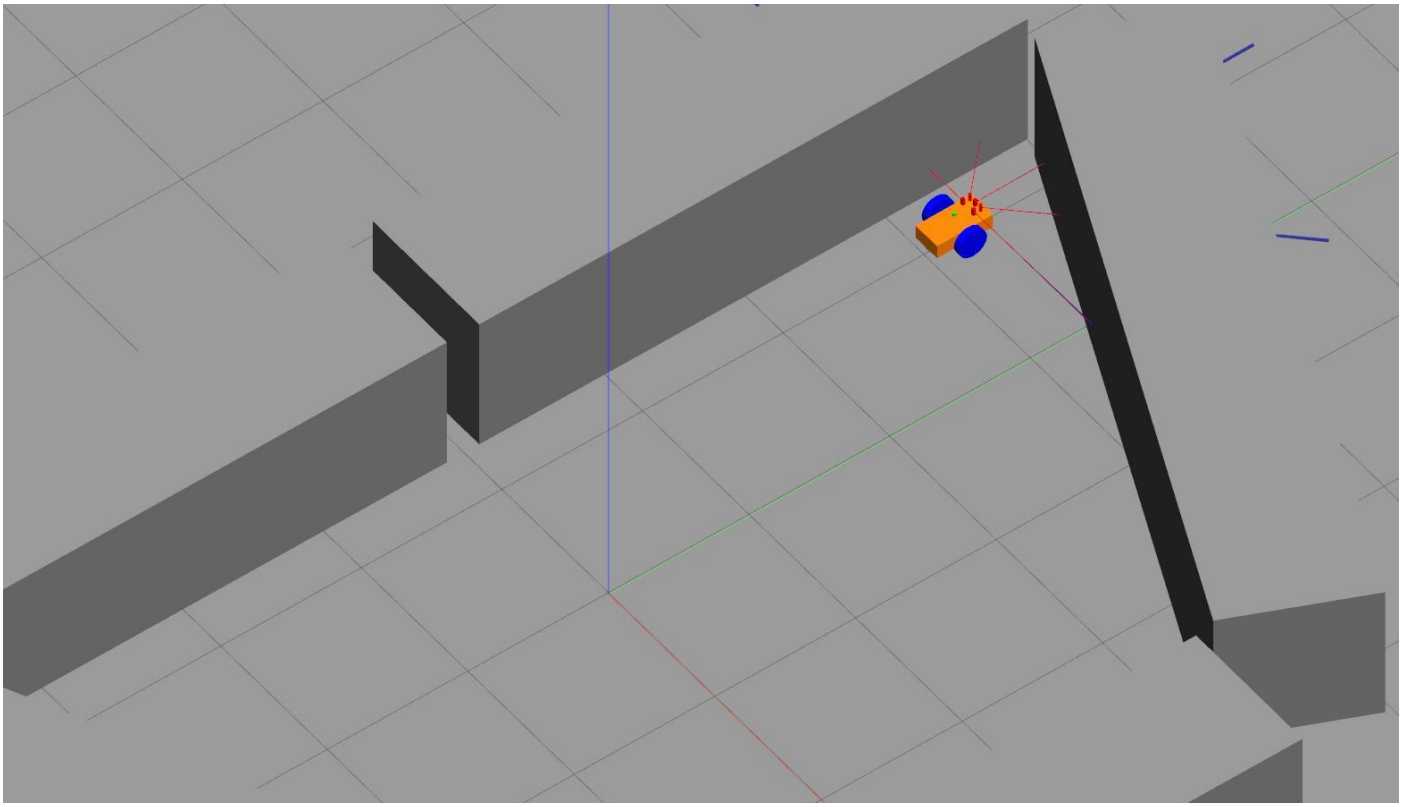




Μπήκε σε σωστή τροχιά



Σταμάτησε το ρομπότ μετά από μία πλήρη περιστροφή



2η Μέθοδος

Αλγόριθμος υλοποίησης και εξισώσεις ελέγχου

Ο δεύτερος τρόπος υλοποίησης του ελέγχου, βασίζεται ξανά στη διάσπαση των εργασιών του ρομπότ σε υποεργασίες – φάσεις, όμως ο προσανατολισμός του ρομπότ καθορίζεται με διαφορετικό τρόπο – χωρίς PD ελεγκτή. Συγκεκριμένα ο αλγόριθμος υλοποίησης βασίζεται στις μετρήσεις των F, L και FL sonars. Όταν οι τιμές και των τριών sonars γίνουν μικρότερες από την επιθυμητή τιμή σημαίνει ότι το ρομπότ προσεγγίζει κάποιο τοίχο. Χρησιμοποιήσαμε τις τιμές και των τριών sonars και όχι μόνο του Front για να εντοπίσουμε εμπόδιο με τη λογική ότι αφού το ρομπότ εντοπίσει εμπόδιο θα αρχίσει να στρίβει και η τελευταία τιμή που θα δει να γίνεται ίση με 0.3m θα είναι του L sonar και εκείνη τη στιγμή θα είναι το ρομπότ παράλληλο με τον τοίχο. Στην συνέχεια έχουμε ορίσει μία φάση που το ρομπότ δεν βλέπει κάποιο εμπόδιο από τους τρεις αισθητήρες που προαναφέραμε, και δημιουργήσαμε και ένα βρόγχο διόρθωσης για τις περιπτώσεις που το ρομπότ αποπροσανατολίζεται προς τη δεξιά πλευρά λόγω του θορύβου (θεωρήσαμε αποδεκτό σφάλμα της τάξης των 5cm για να ξεκινήσει το ρομπότ να διορθώνει).

Στην πρώτη φάση (το ρομπότ δεν βλέπει εμπόδιο) ελέγχονται τα εξής:

- Έχει αρχικά οριστεί μία τελική γραμμική και γωνιακή ταχύτητα ίση με 0.2 m/s και rad/s (κατώφλι) και προσεγγίζονται με τις συνεχείς συναρτήσεις όπως ακριβώς έχει εξηγηθεί και στην πρώτη μέθοδο. Στην φάση αυτή μένουμε μέχρι το ρομπότ να δει κάποιο εμπόδιο.

- Ελέγχεται αν η προηγούμενη φάση που βρισκόταν το ρομπότ ήταν η φάση 2 (βλέπει εμπόδιο). Αυτό σημαίνει ότι το ρομπότ είδε κάποιο εμπόδιο, έκανε τις ενέργειες που θα εξηγηθούν παρακάτω στην φάση 2 -συνοπτικά: επιβράδυνση γραμμικής ταχύτητας, αύξηση αρνητικής (ως προς Z) γωνιακής ταχύτητας (δεξιά στροφή) - και πλέον δεν βλέπει εμπόδιο (εφόσον τώρα είμαστε στην φάση 1). Συνεπώς θα πρέπει να γίνει ομαλή επιβράδυνση της αρνητικής (ως προς Z) γωνιακής ταχύτητας που αναπτύχθηκε για την πραγματοποίηση της περιστροφής, μέχρι αυτή να γίνει μηδενική.

- Ο βρόγχος διόρθωσης εκτελείται όταν η απόσταση που μετράει ο αριστερός αισθητήρας (αυτός που βλέπει το εμπόδιο – τοίχο) του ρομπότ από το εμπόδιο είναι μεγαλύτερη της σταθερής που έχουμε ορίσει ώστε να βρίσκεται παράλληλα σε αυτό. Έτσι όπως και προηγουμένως, επιταχύνουμε ομαλά την θετική (ως προς Z) γωνιακή ταχύτητα του ρομπότ μέχρι να φτάσει την επιθυμητή και αυτό να στρίψει ελαφρώς ώστε να βρεθεί παράλληλα στον τοίχο. Κάθε φορά ελέγχω αν σε προηγούμενη επανάληψη του αλγορίθμου εκτελέστηκε ο βρόγχος ενώ στην επανάληψη που πραγματοποιείται τώρα δεν εκτελέστηκε. Αυτό σημαίνει ότι η απόσταση του ρομπότ από τον τοίχο έφτασε την επιθυμητή (είναι παράλληλα) και πρέπει να επιβραδύνουμε την ομαλά θετική (ως προς Z) γωνιακή ταχύτητα μέχρι να μηδενιστεί ώστε να σταματήσει η αριστερή στροφή.

Στην δεύτερη φάση του αλγορίθμου ελέγχονται τα εξής :

- Εφόσον βλέπω εμπόδιο σε απόσταση μικρότερη μίας ορισμένης τιμής, πρέπει να ξεκινήσω να επιβραδύνω ομαλά την γραμμική ταχύτητα του ρομπότ μέχρι να μηδενιστεί.
- Επίσης θέλουμε να ξεκινήσουμε να επιταχύνουμε αρνητικά (ως προς Z) την γωνιακή ταχύτητα του ρομπότ ώστε να πραγματοποιηθεί δεξιά στροφή για αποφυγή του εμποδίου. Αυτό επιτυγχάνεται και πάλι με τις συνεχείς συναρτήσεις ταχύτητας και επιτάχυνσης που έχουμε

ορίσει. Η επιτάχυνση της γωνιακής ταχύτητας σταματάει όταν αυτή γίνει ίση με 0.2 rad/s όπως εξηγήθηκε παραπάνω.

Τέλος όταν το ρομπότ έχει σχεδόν εκτελέσει μία πλήρη περιστροφή στο περιβάλλον προσομοίωσης, επιβραδύνουμε ομαλά την ταχύτητα του μέχρι αυτή να γίνει μηδενική (γραμμική και γωνιακή), όπου και το ρομπότ σταματάει πλέον την κίνηση του.

Όλες οι συναρτήσεις που χρησιμοποιήθηκαν για τη συνέχεια των ταχυτήτων είναι ίδιες με αυτές που εξηγήθηκαν στην πρώτη μέθοδο. Προφανώς οι εξισώσεις ελέγχου αφορούν μόνο την επιθυμητή ταχύτητα και επιτάχυνση που παίρνουμε ανάλογα με τη φάση στην οποία βρισκόμαστε. Θεωρούμε ότι οι ταχύτητες που δίνουμε μεταφράζονται σωστά. Να τονίσουμε ότι άμα είχαμε τις πραγματικές τιμές της ταχύτητας και επιτάχυνσης (χρήση IMU) θα μπορούσαμε να κάνουμε και έλεγχο ταχύτητας. Θα πρέπει να αναφέρουμε πως σε αυτή τη μέθοδο πετυχαίνουμε τη συνέχεια των ταχυτήτων και επιταχύνσεων αλλά όχι απόλυτα. Για να έχουμε απόλυτη συνέχεια θα πρέπει να ολοκληρώνεται το διάστημα των 0.5s που απαιτείται για να επιταχύνει (ή να επιβραδύνει) πλήρως το ρομπότ πριν μπούμε σε μία άλλη φάση στην οποία η κίνηση περιγράφεται από διαφορετικές συναρτήσεις. Σε περίπτωση που δεν ολοκληρωθεί το διάστημα των 0.5s θα μπούμε κατευθείαν στην επόμενη φάση με την τελική επιθυμητή ταχύτητα (πρακτικά δεν θα επιταχύνουμε ή επιβραδύνουμε). Τρέχοντας το rostopic echo /cmd_vel παρατηρήσαμε ότι αυτό συμβαίνει πολύ σπάνια (συνήθως όταν μπαίνει από το βρόγχο διόρθωσης στη φάση 2) και αυτό οφείλεται στο ότι επιβάλλαμε στο ρομπότ ένα πολύ μικρό χρονικό διάστημα (0.5s) για να επιταχύνει. Θα μπορούσαμε να αποφύγουμε εντελώς αυτή την ασυνέχεια με τη χρήση PD ελεγκτή (καθώς ο PD ελεγκτής θα μεταβάλλει εκθετικά την ταχύτητα) ή υπολογίζοντας δυναμικά καινούργιες κάθε φορά συναρτήσεις συνέχειας ανάλογα με τις εκάστοτε αρχικές συνθήκες ταχύτητας και επιτάχυνσης.

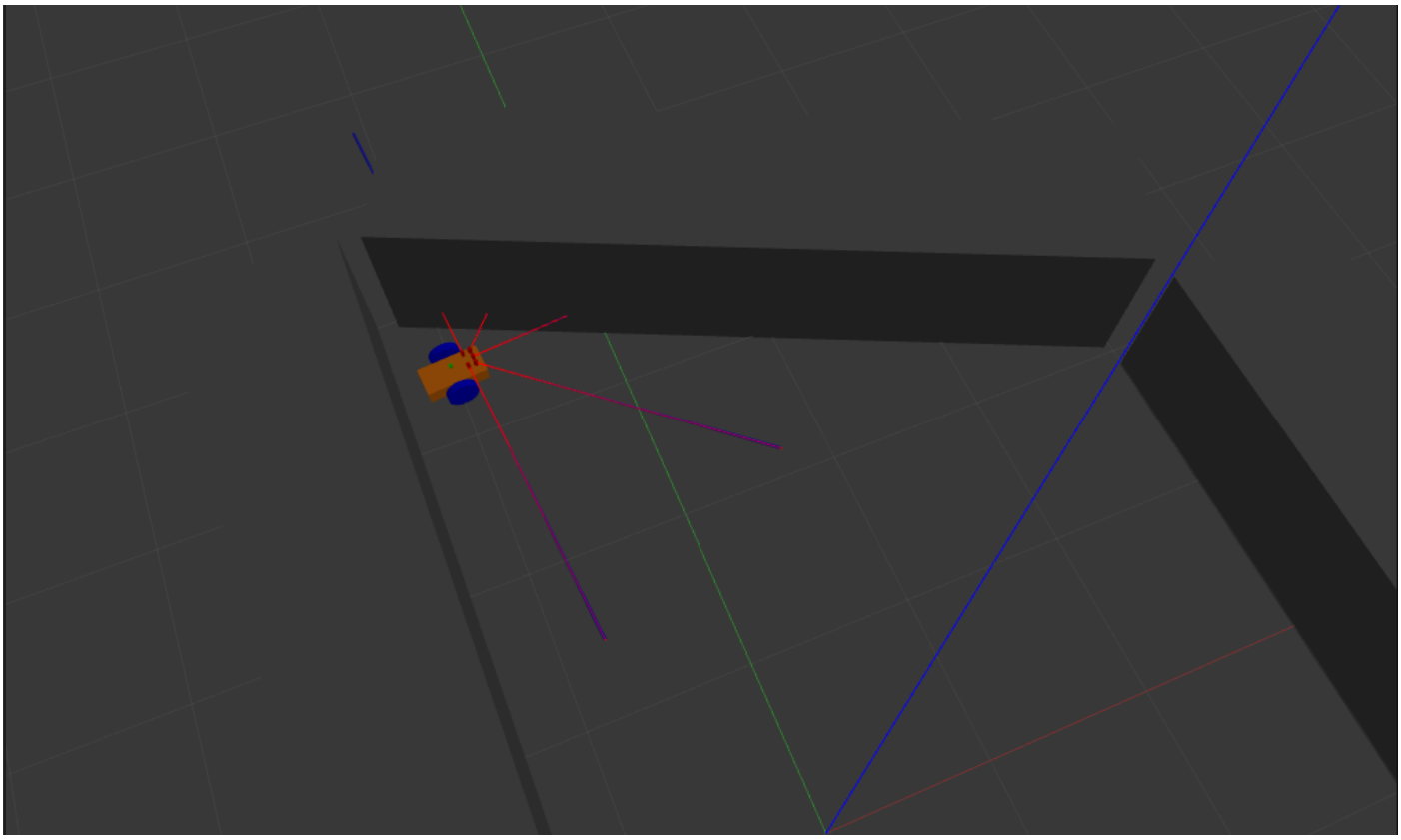
Εξισώσεις ελέγχου: velocity.linear.x = v_d

velocity.angular.z = a_d

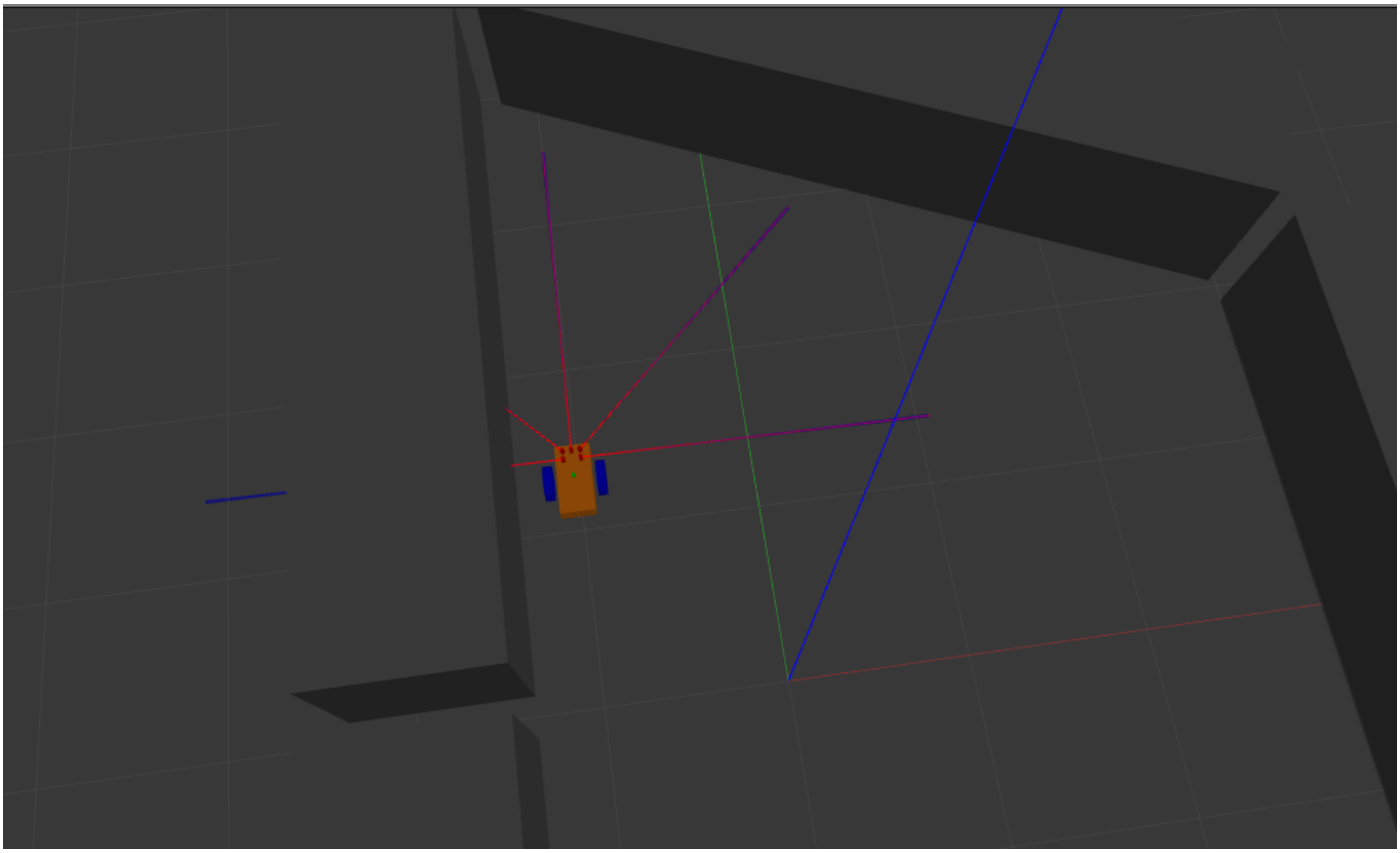
Παρακάτω βλέπουμε μερικά στιγμιότυπα από την εκτέλεση της περιφοράς του ρομπότ στο χώρο που επιδεικνύουν την αποτελεσματικότητα του αλγορίθμου:

<u>Σημείωση: Η προσομοίωση διαρκεί περίπου 6 λεπτά (Real time).</u>

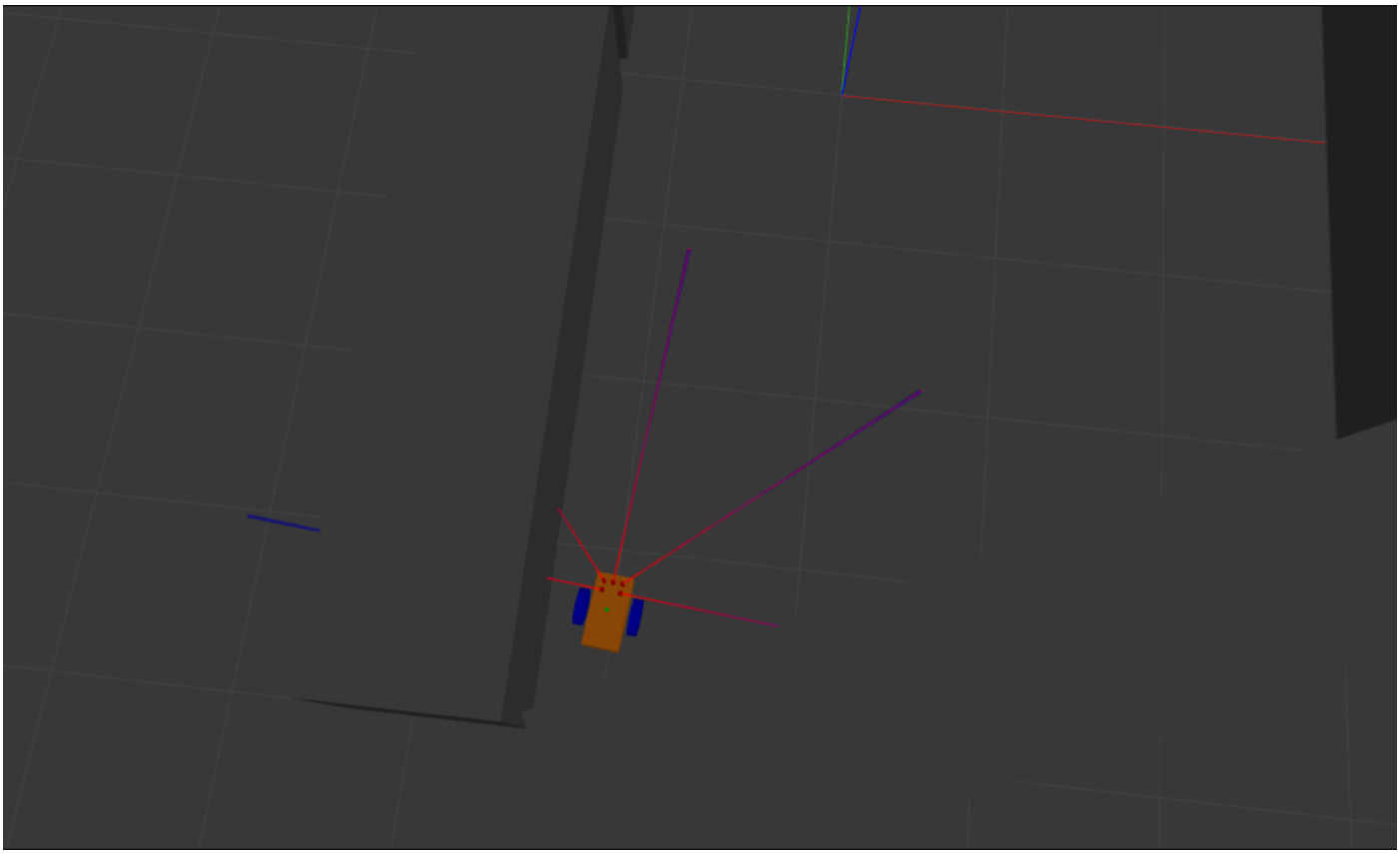
Περιστροφή του ρομπότ για την αποφυγή των εμποδίων (φάση 2):



Κίνηση παράλληλη με τον τοίχο (φάση 1):

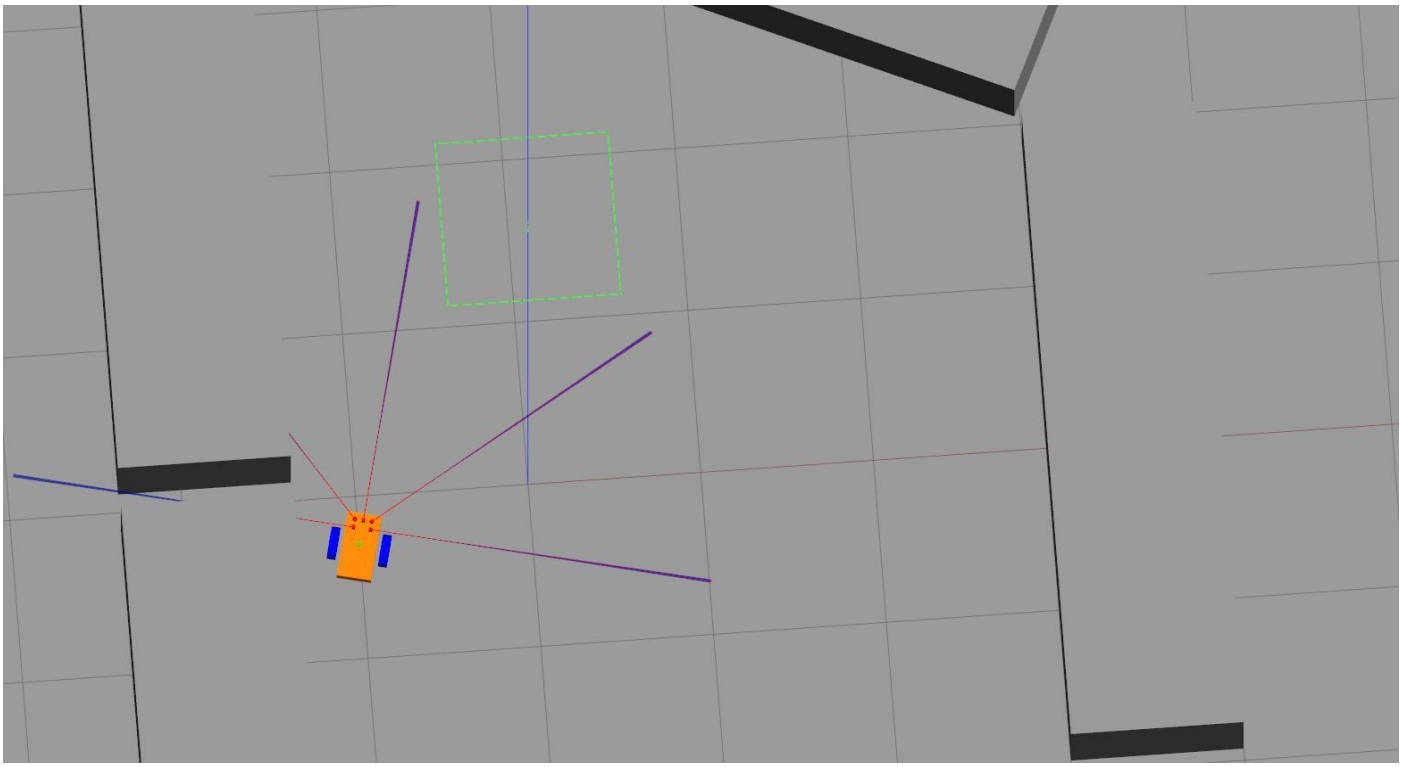


Αποπροσανατολισμός του ρομπότ (εκτέλεση του βρόγχου διόρθωσης)

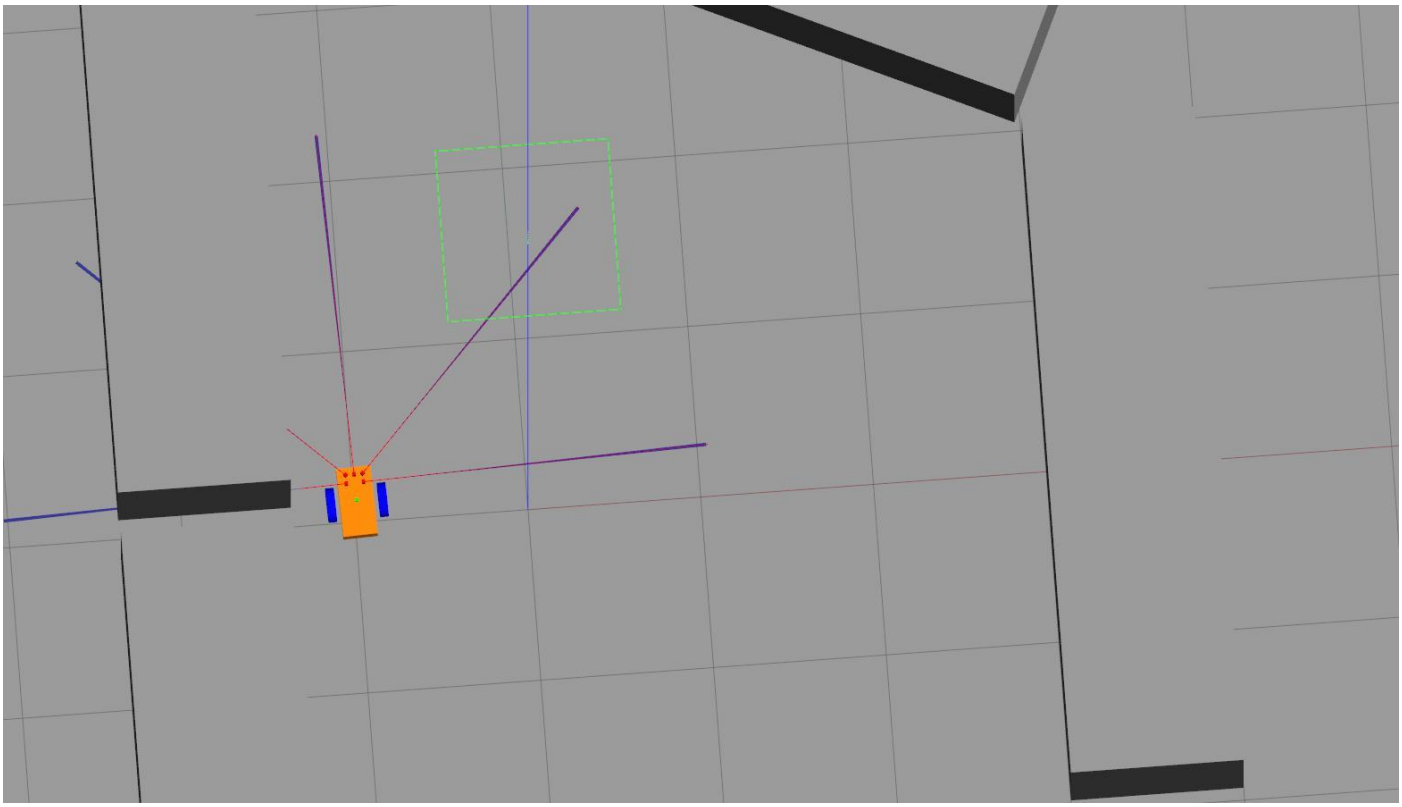


Συμπεριφορά του ρομπότ στο σημείο που παίρνει λάθος μετρήσεις:

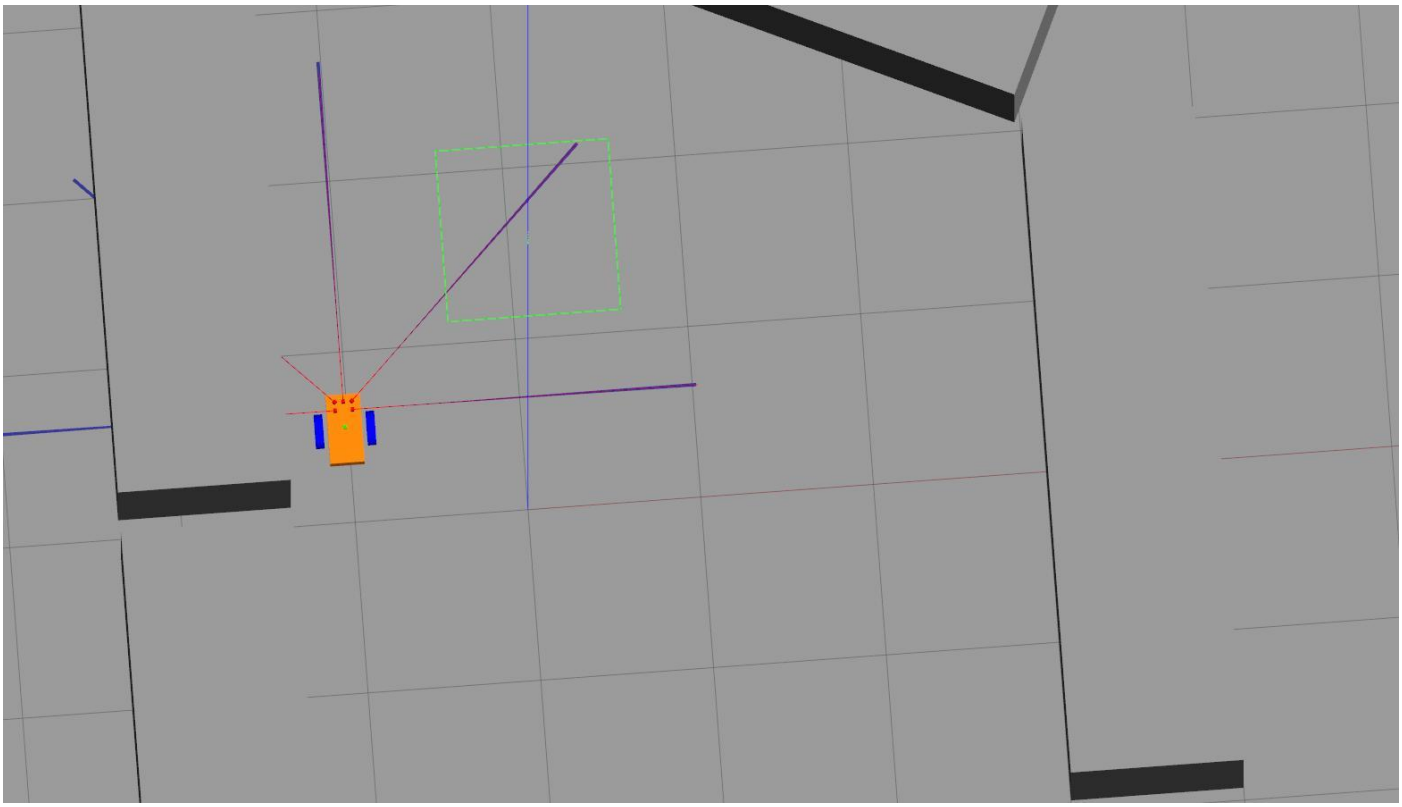
Αποπροσανατολισμός από λάθος μέτρηση του L sonar



Διορθώνει



Μπήκε σε σωστή τροχιά



Σταμάτησε το ρομπότ μετά από μία πλήρη περιστροφή