Ρομποτική ΙΙ

Εξαμηνιαία Εργασία 2

Μέρος Α

Αυτοκινούμενα ρομπότ: Παρακολούθηση εμποδίου
(Mobile robots: Wall Following)

Αντωνίου Κωνσταντίνος
Α.Μ.: 03116169
kostas.an2016@gmail.com

Μετζάκης Ιωάννης
Α.Μ.: 03116202
johnmetzakis@gmail.com

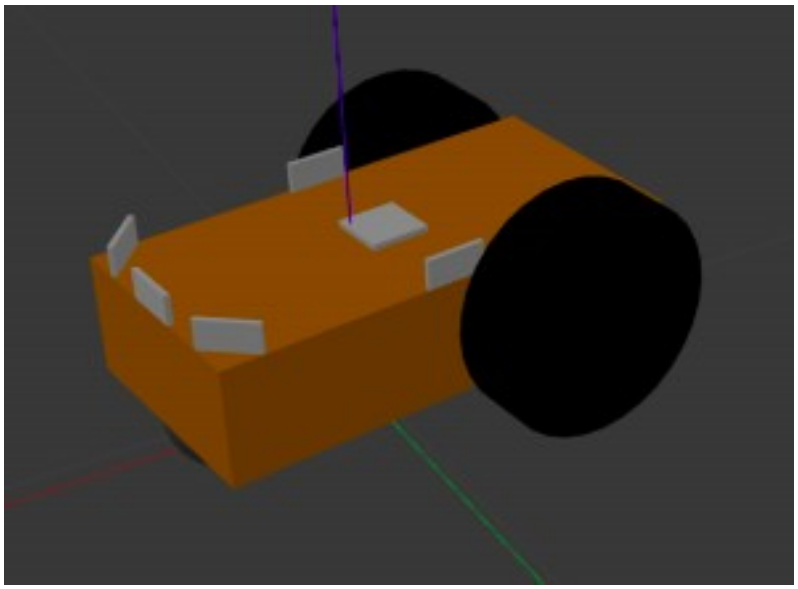
ΣΗΜΜΥ 8ο

Εισαγωγή:

Σκοπός του 1ου μέρους της 2ης εξαμηνιαίας εργασίας είναι η αποφυγή εμποδίων, καθώς και η παρακολούθησή τους. Συγκεκριμένα, το ρομπότ καλείται να εκτελεί τα παρακάτω:

- Να ακολουθεί (παράλληλα) έναν τοίχο
- Να κρατάει σταθερή απόσταση από αυτόν.

Το ρομπότ που καλούμαστε να χρησιμοποιήσουμε φαίνεται παρακάτω:



Διάταξη:

Το ρομπότ που θα χρησιμοποιήσουμε είναι διαφορικής οδήγησης με 2 τροχούς. Είναι εξοπλισμένο με 6 αισθητήρες εκ των οποίων:

- Οι πέντε είναι τύπου υπερήχων sonar οι οποίοι μετρούν απόσταση απο εμπόδια
- Ο ένας είναι τύπου IMU (internal measurement unit) 9 βαθμών ελευθερίας, ο οποίος μετράει στροφικές ταχύτητες, γραμμικές επιταχύνσεις, καθώς και περιστροφή γύρω από κάθε άξονα.

Τα πλαίσια αναφοράς και οι αποστάσεις φαίνονται στην παρακάτω φωτογραφία:

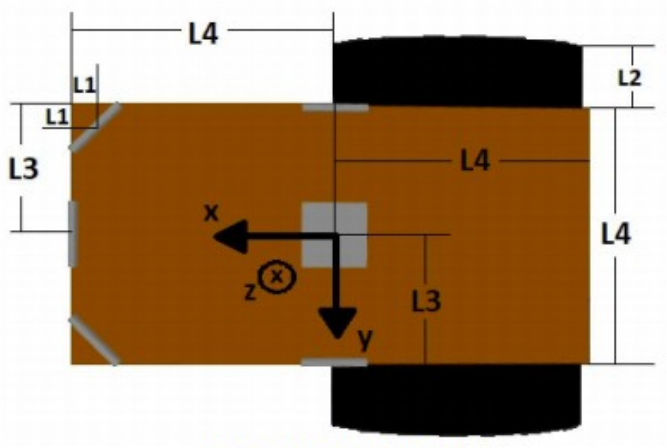
Οι τιμές των μεγεθών είναι:

$L1 = 0.018\,m$

$L2 = 0.05\,m$

$L3 = 0.1\,m$

$L4 = 0.2\,m$



Εικόνα 2 - Scheme Design

Διάταξη αρχικοποίησης και φορά περιστροφής:

Ο αρχικός προσανατολισμός (περιστροφή ως προς τον άξονα z)που θέλουμε να έχουμε ορίζεται ως:

$$angle = mod\,(X,\pi)$$

όπου X=X1 + X2 = 9 + 2 = 11,
και επομένως:

$$\boldsymbol{angle = 1.57522204}$$

Επίσης, το X είναι περιττός αριθμός. Συνεπώς, η φορά περιστροφής που καλούμαστε να ακολουθήσουμε είναι **CCW (Counter Clock Wise).**

Θεωρητική Ανάλυση:

Η ανάλυση μας χωρίζεται σε 3 στάδια:

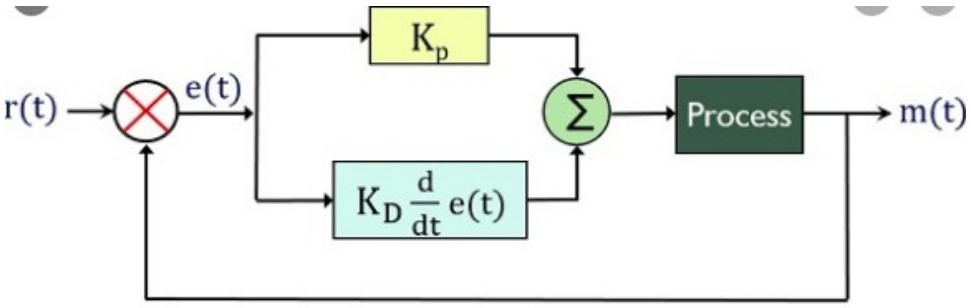
1. Κίνηση προς τον τοίχο απο την θέση εκκίνησης (αρχική κίνηση).
2. PD ελεγκτής και κίνηση παράλληλα στον τοίχο.
3. Σταμάτημα και περιστροφή του ρομπότ όταν αντιμετωπίζει τις γωνίες που σχηματίζουν οι τοίχοι μεταξύ τους.

Στάδιο 1ο:

Για να πλησιάσουμε τον τοίχο για πρώτη φορά θέτουμε γραμμική ταχύτητα ως προς τον άξονα x του ρομπότ ίση με 0.2m/s και γωνιακή ταχύτητα -0.02 rad/s ως προς τον άξονα z. Η συγκεκριμένη διαδικασία εκτελείται μόνο μια φορά και κατά συνέπεια, το κομμάτι κώδικα που την υλοποιεί δεν ξαναχρησιμοποιείται (χρήση της μεταβλητής first_time). Η κίνηση αυτή σταματά όταν είτε ο front είτε ο front_right αισθητήρας "δουν" απόσταση μικρότερη απο 0.33m. Η απόσταση αυτή επιλέχτηκε πειραματικά, καθώς μας έδινε μικρότερο σφάλμα.

Στάδιο 2ο:

Το γενικό σχήμα στο οποίο βασίζεται ο pd controller φαίνεται παρακάτω:



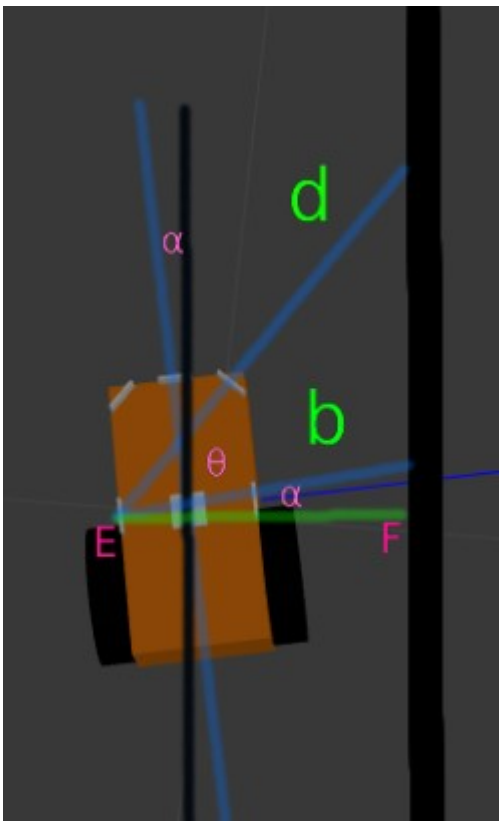
Το σήμα u, το οποίο μπαίνει στο process έχει την μορφή:

$$u = Kp*(xd-xr) + Kd* (xd'-xr')$$

όπου *xd* είναι η επιθυμητή θέση, *xr* η πραγματική θέση, *xd'* η επιθυμητή ταχύτητα και *xr'* η πραγματική ταχύτητα. Ορίζουμε το σφάλμα ως $e = xd - xr$ και αντίστοιχα $e' = xd' - xr'$. Την επιθυμητή απόσταση την ορίσαμε ίση με 0.3m, μετρώντας απο το δεξί sonar ή 0.4m από το κέντρο του ρομπότ.

Ο pd ελεγκτής συνδιάζει έναν αναλογικό έλεγχο (Kp) και έναν διαφορικό έλεγχο (Kd). Αυξάνοντας το Kp πετυχαίνουμε ελάττωση του χρόνου ανύψωσης καθώς και το σφάλμα στην μόνιμη κατάσταση. Όσο και αν αυξλησουμε το Kp δεν θα πετύχουμε ποτέ την εξάλειψη του μόνιμου σφάλματος. Γι' αυτό συνδιάζουμε τον αναλογικό ελεγκτή με έναν διαφορικό. Η αύξηση του Kd προκαλεί την μείωση του overshoot και μείωση του μόνιμου σφάλματος σε επίπεδα που δεν θα πετυχαίναμε ποτέ μόνο με τον αναλογικό ελεγκτή. Ωστόσο, απαιραίτητο είναι να συνδιαστούν σωστά οι τιμές των Kp και Kd ώστε να επιτύχουμε το καλύτερο δυνατό αποτέλεσμα.

Το σχήμα στο οποίο βασίστηκε η ανάλυση του ελεγκτή μας φαίνεται παρακάτω:



Αρχικά, πρέπει να υπολογίσουμε την γωνία α, η οποία στην ουσία εκφράζει την “απόκλιση" του ρομπότ από την ευθεία που είναι παράλληλη στον τοίχο (μαύρη ευθεία στο σχήμα).

Η γωνία θ που σχηματίζεται μεταξύ των b, d είναι 45 μοίρες, αφού αυτά τα τμήματα σχηματίζουν ένα ισοσκελές τρίγωνο. Η γωνία α υπολογίζεται απο τον τύπο:

$$\alpha = \mathrm{Arctan}\left[\frac{d\cos\theta - b}{d\sin\theta}\right]$$

Τα τμήματα d, b είναι οι μετρήσεις του μπροστά δεξιού και του δεξιού sonar αντίστοιχα, αναγόμενες στο αριστερό sonar. Επειδή επιθυμούμε σταθερή απόσταση **0.3m** από τον τοίχο, μετρούμενη απο το δεξί sonar, για να αναγάγουμε τις αποστάσεις του μπροστά δεξιού και του δεξιού sonar στη θέση του αριστερού sonar εκτελούμε τις παρακάτω εντολές:

```
real_r = sonar_right + 0.2
real_fr = sonar_front_right + (np.sqrt(2)*0.2) - np.sqrt(2)*0.018
```

όπου:

-για το real_right sonar απλά προσθέτουμε απόσταση 0.2m η οποία είναι το πλάτος του ρομπότ

-για το real_front_sonar παίρνουμε την μέτρηση απο το front_right sonar και προσθέτουμε την απόσταση απο το αριστερό sonar μέχρι την γωνία του ρομπότ. Έπειτα, αφαιρούμε την απόσταση απο το front_right sonar μέχρι την γωνία του ρομπότ, καθώς την είχαμε συνυπολογίσει 2 φορές.

Χρησιμοποιούμε σαν είσοδο του ελεγκτή μας την απόσταση EF η οποία ορίζεται ως εξής:

$$EF = cos(a) * real\_r$$

όπου το real_r φαίνεται στην παραπάνω φωτογραφία.

Ορίζουμε το σφάλμα ως :

$$error = 0.5 - EF$$

όπου 0.5m είναι η επιθυμητή απόσταση απο το αριστερό sonar μέχρι τον τοίχο.

Έχοντας πλέον το σφάλμα και ξέροντας ότι θέλουμε να επηρεάσουμε με τον ελεγκτή την γωνιακή ταχύτητα στον άξονα z (αφού θέλουμε το ρομπότ να στρίβει κατάλληλα ώστε να κρατά σταθερή απόσταση), έχουμε:

$$\mathbf{self.velocity.angular.z = -Kp*error - Kd*(error - previous\_error)}$$

Ταυτόχρονα, δίνουμε και γραμμική ταχύτητα στον x άξονα ίση με 0.2m/s.

Επιλέγουμε $Kp= 14$ και $Kd= 16$, καθώς για αυτές τις τιμές παρατηρήσαμε (πειραματικά) τον καλύτερο συνδυασμό σφάλματος μόνιμης κατάστασης και overshoot.

Για να ανιχνεύσουμε πότε πρέπει να είμαστε στο στάδιο ελέγχου με pd ελεγκτή ακολουθούμε την παρακάτω σχέση:

```
if (sonar_right < sonar_front+0.1 and sonar_front_right<sonar_front+0.1 and
    sonar_front_left >= 0.6):
```

Η λογική πίσω απο αυτή τη συνθήκη είναι η εξής:

Για να παραμείνουμε στο στάδιο 2, πρέπει ο τοίχος να παραμένει ευθεία και να μην έχει στροφές, πράγμα το οποίο σημαίνει ότι η απόσταση που βλέπει το δεξί sonar θα είναι μικρότερη από την απόσταση που βλέπει το μπροστινό sonar.
Επίσης, με την ίδια λογική, η απόσταση που θα βλέπει το μπροστά δεξία sonar θα είναι επίσης μικρότερη απο την απόσταση που βλέπει το δεξί sonar. Η προσθήκη του 0.1m έγινε πειραματικά, καθώς με αυτό το τρόπο το ρομπότ πλησίαζε πιο πολύ στην στροφή, και κατά συνέπεια “την έπαιρνε” πιο κλειστά.
Τέλος, προσθέσαμε και τη συνθήκη για το μπροστά αριστερά sonar, έτσι ώστε να βλέπουμε εγκαίρως αν πλησιάζουμε σε κλειστή στροφή, για πιο άμεση αντιμετώπιση. Η τιμή του 0.6m έγινε επίσης πειραματικά.

Στάδιο 3:

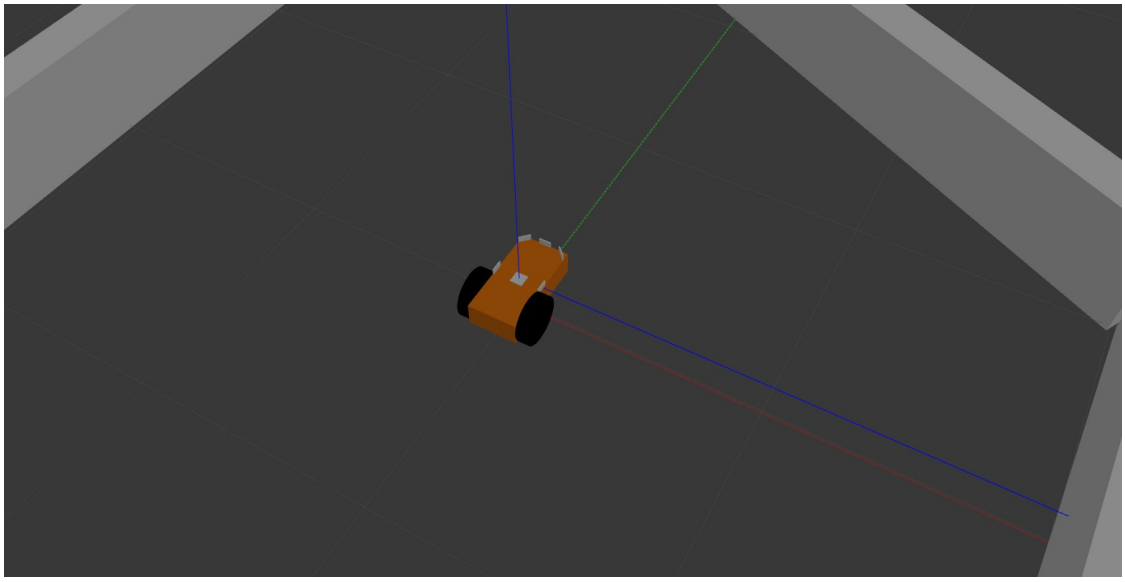
Το τρίτο στάδιο αφορά αποκλειστικά τη στροφή του ρομπότ. Η συνθήκη για να μπει σε αυτό το στάδιο είναι απλά η άρνηση της συνθήκης για να είναι στο στάδιο 2, καθώς το ρομπότ είτε θα πρέπει να στρίψει είτε να ακολουθησεί μια ευθεία παράλληλα στον τοίχο. Επομένως, σε αυτό το κομμάτι απλά δίνουμε στο ρομπότ μηδενική γραμμική ταχύτητα στον άξονα x (καθώς θέλουμε να σταματάει και μετά να στρίβει) και γωνιακή ταχύτητα -0.4rad/sec στον άξονα z.

Επειδή ο έλεγχος των συνθηκών γίνεται συνεχώς, όταν έρθει πάλι σε ισχύ η σχέση για να μπούμε στο στάδιο 2, τότε θα τεθεί και πάλι σε εφαρμογή ο PD ελεγκτής, προσπαθώντας να διορθώσει το σφάλμα που προήλθε απο το στάδιο 3.

Προσομοίωση:

Εκκινώντας το gazebo, το ρομπότ εμφανίζεται πάνω στη διεύθυνση του άξονα y (αντί του άξονα x όπου θα έβγαινε αν η angle init ήταν ίση με 0), κάτι το οποίο είναι απόλυτα λογικό, μιας και η γωνία αρχικοποίησης που υπολογίσαμε βάσει των αριθμών μητρώων μας ισούται με 1.57522204rad, δηλαδή π/2rad.

Παρακάτω, το στιγμιότυπο από την αρχικοποίηση του ρομπότ:



Στη συνέχεια, ξεκινάμε την προσομοίωση, εκτελώντας το αρχείο "*mobile_partA.launch"* . Ο συνολικός χρόνος που εκτελεί μία πλήρη περιστροφή, είναι περίπου 75sec, ενώ χρειάζεται ακόμα περίπου 4-5sec για να εντοπίσει αρχικά τον πρώτο τοίχο. (Σε εμάς η προσομοίωση διαρκούσε άπειρο χρόνο, λόγω χαμηλής απόδοσης υπολογιστή, και real time factor < 0.4 -έπιανε ακόμα και 0.1..- αλλά μέσω terminal και κώδικα, υπολογίσαμε τον ακριβή χρόνο).

<u>Στάδιο 1:</u>

Όπως αναφέραμε και προηγουμένως, στο στάδιο αυτό, το ρομπότ κινείται ευθεία έως ότου συναντήσει έναν τοίχο, έχοντας γραμμική ταχύτητα ίση με 0.4m/s, καθώς και μία ελάχιστη γωνιακή ταχύτητα ίση με -0.02rad/s, έτσι ώστε τη στιγμή που θα συναντήσει τον τοίχο να έχει ήδη πάρει μία μικρή κλίση, διευκολύνοντας την αρχική είσοδο στο 2^{ο} στάδιο και τον pd ελεγκτή.

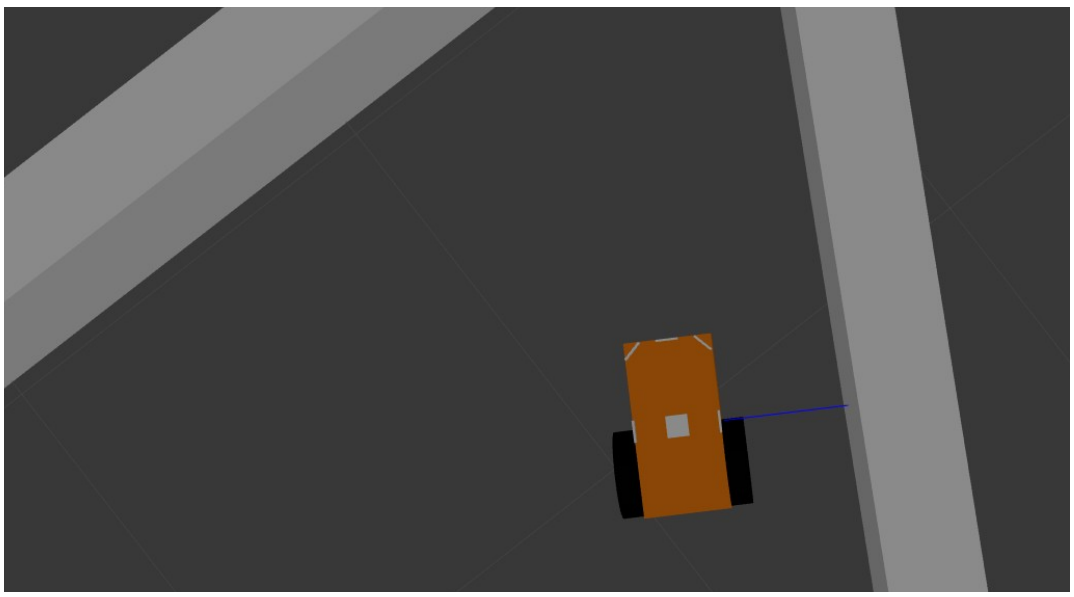
Όπως φαίνεται, η γωνιακή ταχύτητα που δίνουμε είναι αρνητική, καθώς και πάλι λόγω των αριθμών μητρώων μας, η περιστροφή που ζητείται να εκτελέσει το ρομπότ μας είναι counterclockwise.

Παρακάτω, φαίνονται μερικά συμπτηγμένα στιγμιότυπα της κίνησης του ρομπότ, από όταν ακόμα “ψάχνει” τον τοίχο, μέχρι και όταν έχει μπεί στον pd ελεγκτή, και έχει σταθεροποιηθεί στην επιθυμητή απόσταση από τον τοίχο, δηλαδή στα 0.3m (κάθετη απόσταση από δεξί sonar) :

Όπως βλέπουμε από την παραπάνω φωτογραφία, και θα δούμε και στη συνέχεια μέσω των γραφικών παραστάσεων διάφορων σημαντικών παραμέτρων, υπάρχει ένα overshoot κατά την είσοδο στον pd ελεγκτή, το οποίο προκαλεί ένα αρχικό σφάλμα, που στη συνέχεια όμως εξομαλύνεται.

Στάδιο 2:

Στο σημείο αυτό, έχουμε ήδη μπει στο 2ο στάδιο, εκτελώντας το task του wall following, και διατηρώντας ταυτόχρονα σταθερή κάθετη απόσταση μεταξύ δεξιού sonar και τοίχου, ίση με **0.3m**, όπως φαίνεται και στο παρακάτω στιγμιότυπο:

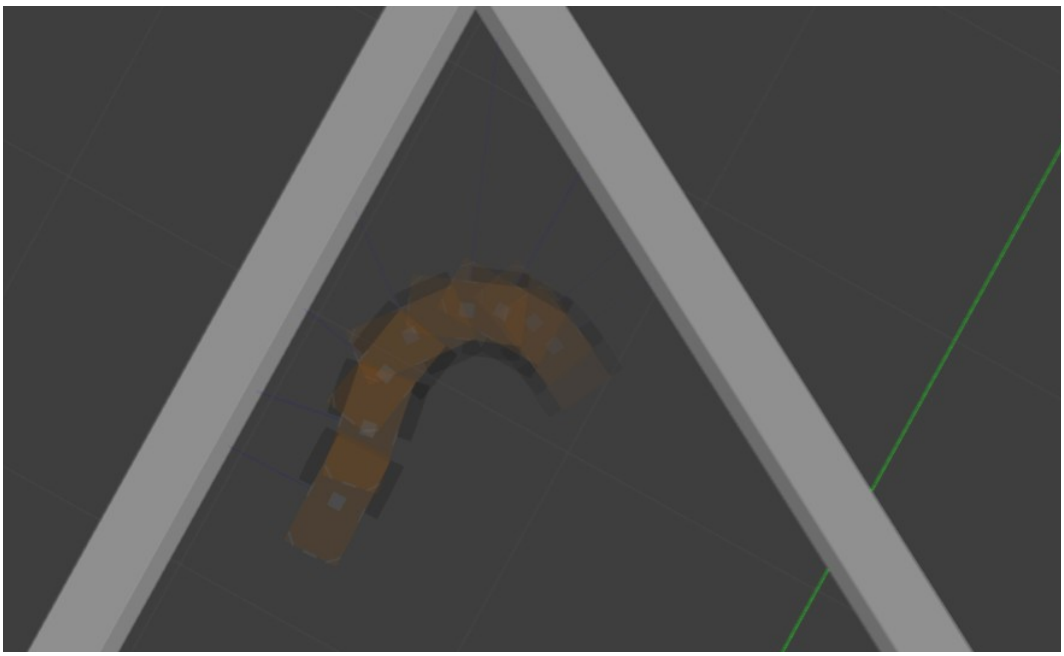


Στάδιο 3:

Στη συνέχεια, μόλις το μπροστά αριστερό sonar “δει” απόσταση μικρότερη από 0.6m (η αντίστροφη συνθήκη είναι προϋπόθεση για να βρίσκεται στο 2^{ο} στάδιο), το ρομπότ εισέρχεται στο 3^{ο} στάδιο, όπου καλείται να μειώσει την γραμμική ταχύτητα σε 0.1m/s και να αυξήσει το μέτρο της γωνιακής ταχύτητας σε 0.4rad/s, έτσι ώστε να στρίψει και να αποφύγει με επιτυχία την πρώτη οξεία γωνία.

Γενικότερα, το ρομπότ εισέρχεται στο 3^{ο} στάδιο μόνο στις δύο οξείες γωνίες, καθώς για τις αμβλείες γωνίες αρκεί ο pd έλεγχος.

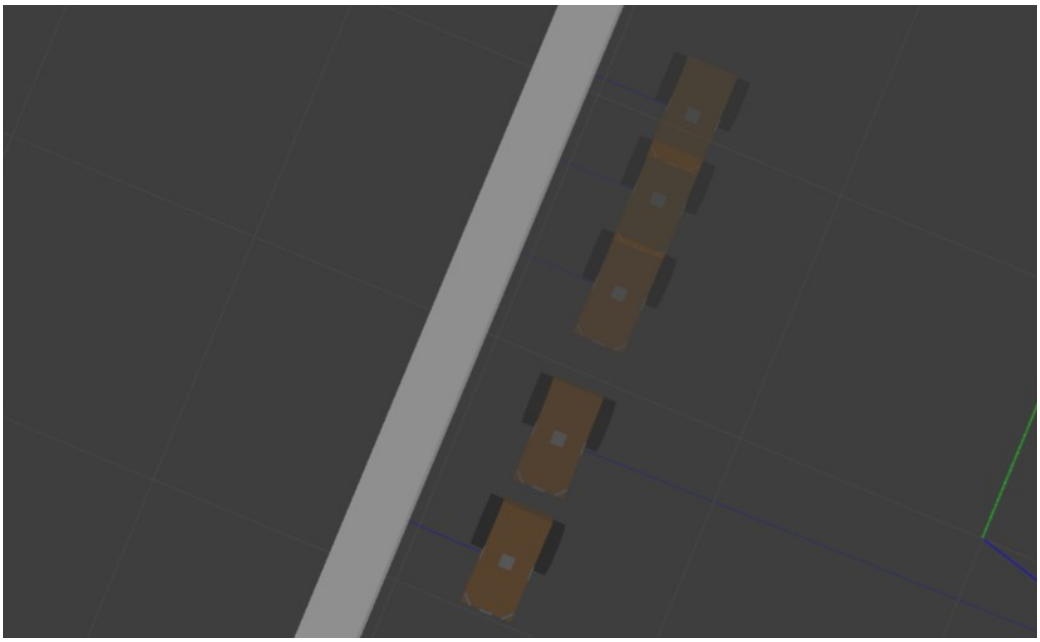
Παρακάτω, βρίσκονται μερικά επίσης συμπτηγμένα στιγμιότυπα της πρώτης οξείας γωνίας, δείχνοντας τη συμπεριφορά του ρομπότ από όταν εντοπίσει τον τοίχο στα αριστερά, μέχρι να σταθεροποιηθεί σε σταθερή απόσταση από τον νέο τοίχο:



Όπως βλέπουμε και εδώ, υπάρχει το ίδιο overshoot κατά την είσοδο στον pd ελεγκτή στο τέλος της στροφής, μέχρι το ρομπότ να σταθεροποιηθεί στη ζητούμενη απόσταση.

Στη συνέχεια, το ρομπότ βρίσκεται στη μεγάλη ευθεία της “πίστας”, όπου μέσω του pd ελεγκτή διατηρείται σταθερή η επιθυμητή απόσταση από τον τοίχο, σε όλη τη διάρκεια της ευθείας.

Παρακάτω, παρουσιάζεται ένα μέρος της κίνησης του ρομπότ σε αυτή την ευθεία:



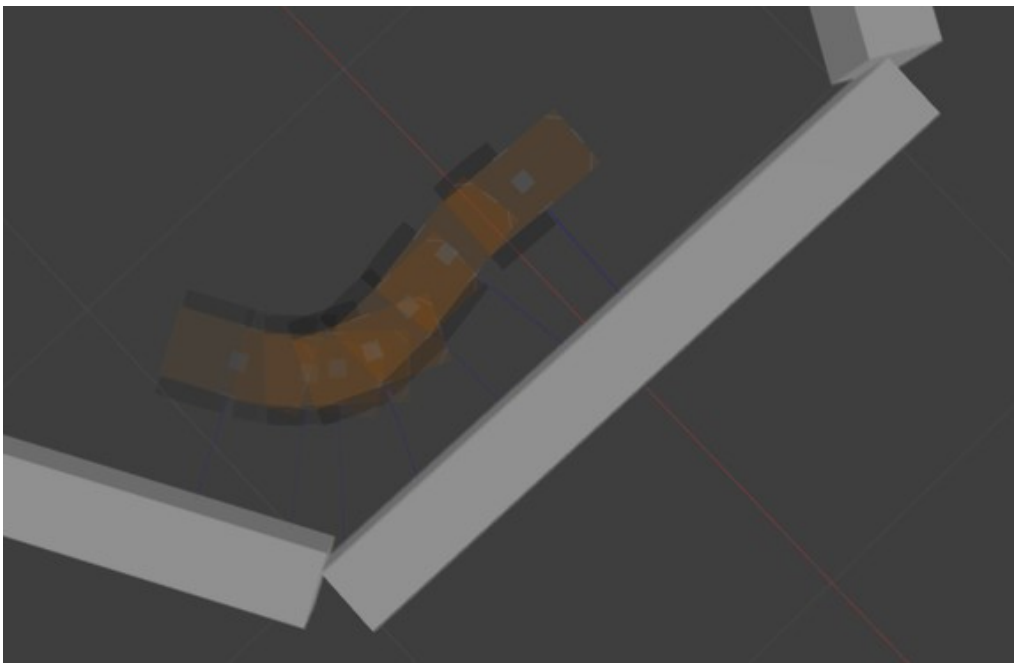
Παρατηρούμε πως ο pd ελεγκτής διατηρεί επιτυχώς το ρομπότ στην επιθυμητή απόσταση, προσπαθώντας συνεχώς να εξισορροπήσει πιθανές αποκλίσεις.

Στην παραπάνω φωτογραφία αυτό φαίνεται ίσως ελάχιστα, στις 2 τελευταίες θέσεις του ρομπότ, όπου ενώ στην προτελευταία πάει να απομακρυνθεί ελαφρώς, στην τελευταία θέση το επαναφέρει στην επιθυμητή απόσταση.

Έπειτα, ακολουθεί η δεύτερη οξεία γωνία, όπου η συμπεριφορά του ρομπότ είναι ίδια με τη συμπεριφορά που είχε στην πρώτη, συνεπώς δε θα δείξουμε κάποιο στιγμιότυπο.

Η τελευταία, άξια αναφοράς, κατάσταση, είναι η αμβλεία γωνία και η συμπεριφορά του ρομπότ όταν τη συναντά. Όπως και πριν, θα δείξουμε τα στιγμιότυπα μόνο της πρώτης, αφού και πάλι το ρομπότ έχει την ίδια συμπεριφορά και στις δύο πανομοιότυπες στροφές.

Παρακάτω βρίσκονται όλα τα στάδια της στροφής:



Δεν παρατηρούμε κάτι διαφορετικό από τις προηγούμενες στροφές, όσον αφορά τη συμπεριφορά του ρομπότ, το αρχικό σφάλμα, και την άμεση επαναφορά στην επιθυμητή απόσταση.

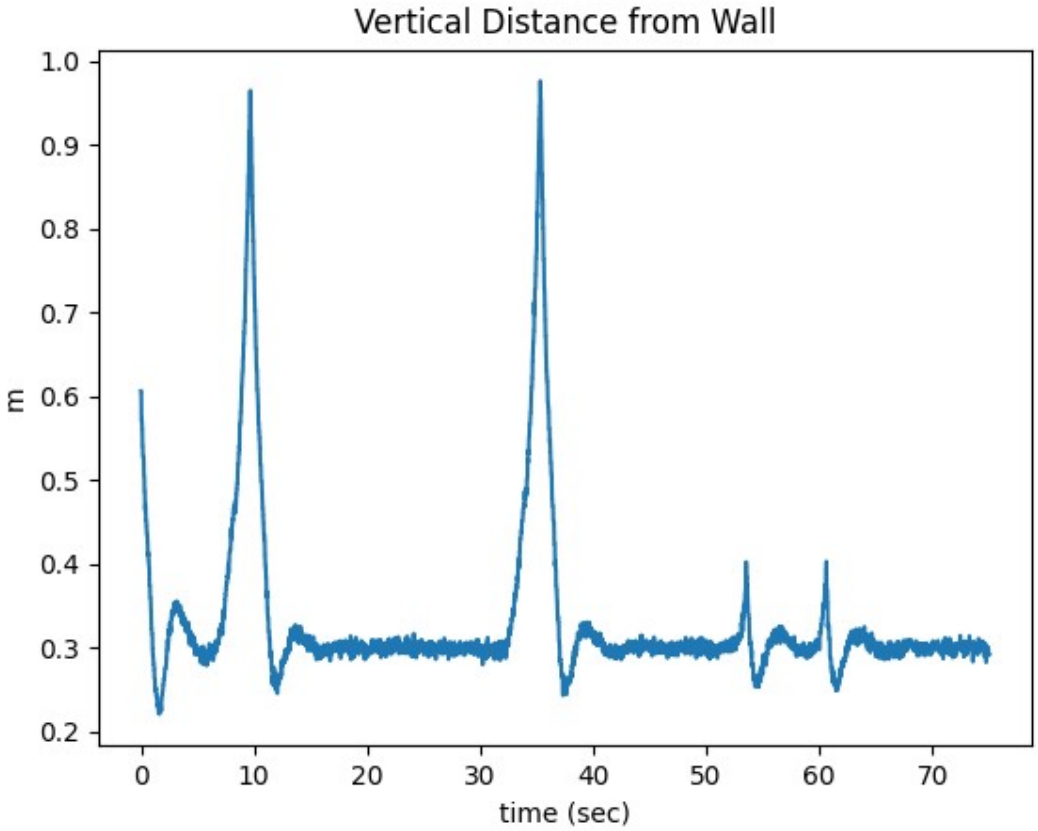
Γραφικές Παραστάσεις:

Θεωρήσαμε περισσότερο ενδιαφέρουσες τις παρακάτω παραμέτρους:

1. Κάθετη απόσταση μεταξύ δεξιού sonar και τοίχου.
2. Γωνία που σχηματίζει το ρομπότ με μια νοητή παράλληλη του τοίχου (0rad συνεπάγεται παράλληλα στον τοίχο).
3. Γραμμική ταχύτητα στον άξονα x.
4. Γωνιακή ταχύτητα στον άξονα z.

Κατά συνέπεια, δημιουργήσαμε τις γραφικές παραστάσεις των προαναφερθέντων παραμέτρων σε συνάρτηση με τον χρόνο, για τη διάρκεια μίας πλήρους περιστροφής.

Αξίζει να σημειωθέι πως η μέτρηση των παραμέτρων αυτών και η αποτύπωσή τους στις γραφικές παραστάσεις, ξεκίνησε αφού ολοκληρώθηκε η αρχική αναζήτηση τοίχου, άρα τη χρονική στιγμή t=0, το ρομπότ έχει μόλις εντοπίσει τοίχο, και αρχίζει το wall following.



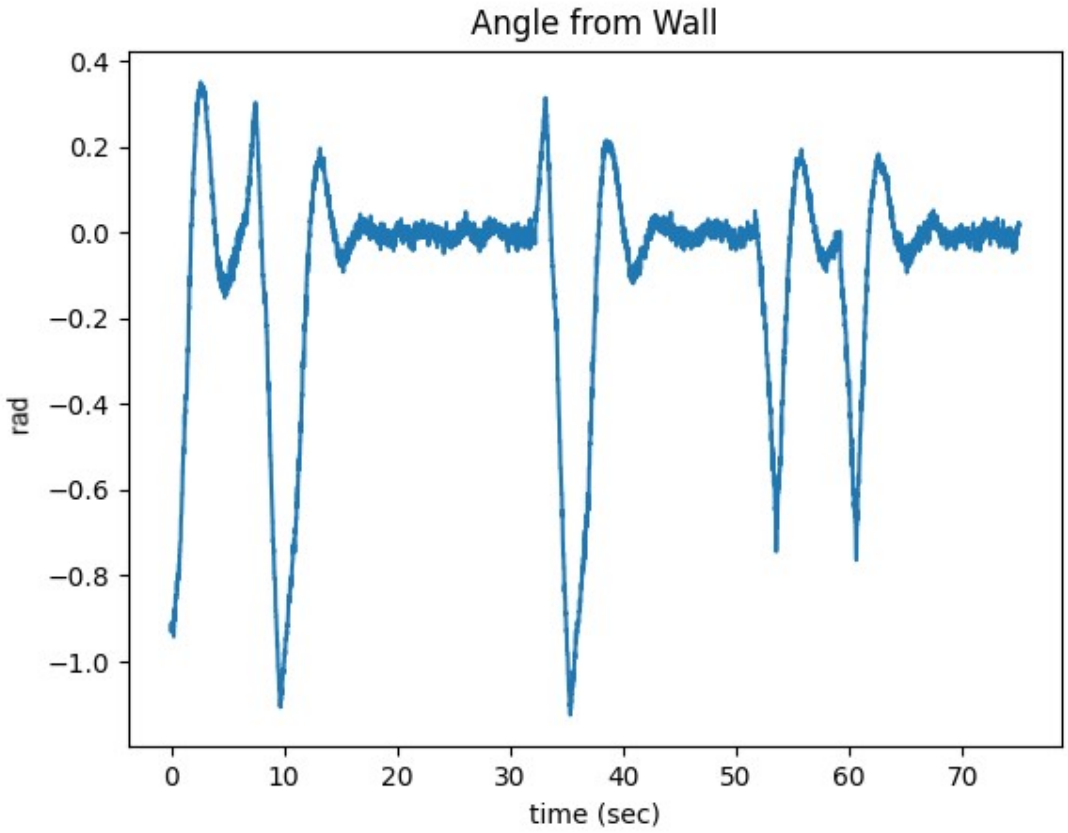
Παρατηρούμε πως υπάρχουν κάποιες αρκετά μεγάλες μεταβολές στην απόσταση από τον τοίχο, οι οποίες προφανώς αντιστοιχούν στις 4 γωνίες.

Συγκεκριμένα, βλέπουμε πως περίπου στα 7-8sec συναντάμε την πρώτη οξεία γωνία και γύρω στα 33-34sec τη δεύτερη, ενώ γύρω στα 52sec και 60sec συναντάμε τις δύο αμβλείες γωνίες.
Επίσης, η μεταβολή στην αρχή οφείλεται στο ότι τη στιγμή που το ρομπότ εντοπίζει τον πρώτο τοίχο (όπου και αρχίζει ο χρόνος μέτρησης των παραμέτρων), έχει μία ήδη σημαντική κλίση.

Συνεπώς, αναμένουμε να δούμε αντίστοιχες μεταβολές στα συγκεκριμένα χρονικά σημεία και στις υπόλοιπες γραφικές (οπότε δε θα τα ξανά - αναλύσουμε στη συνέχεια).

Ωστόσο, ενδιαφέρον παρουσιάζουν τα διαστήματα μετά από κάθε απότομη μεταβολή, όπου φαίνεται ξεκάθαρα το overshoot και η σταδιακή εξομάλυνση.

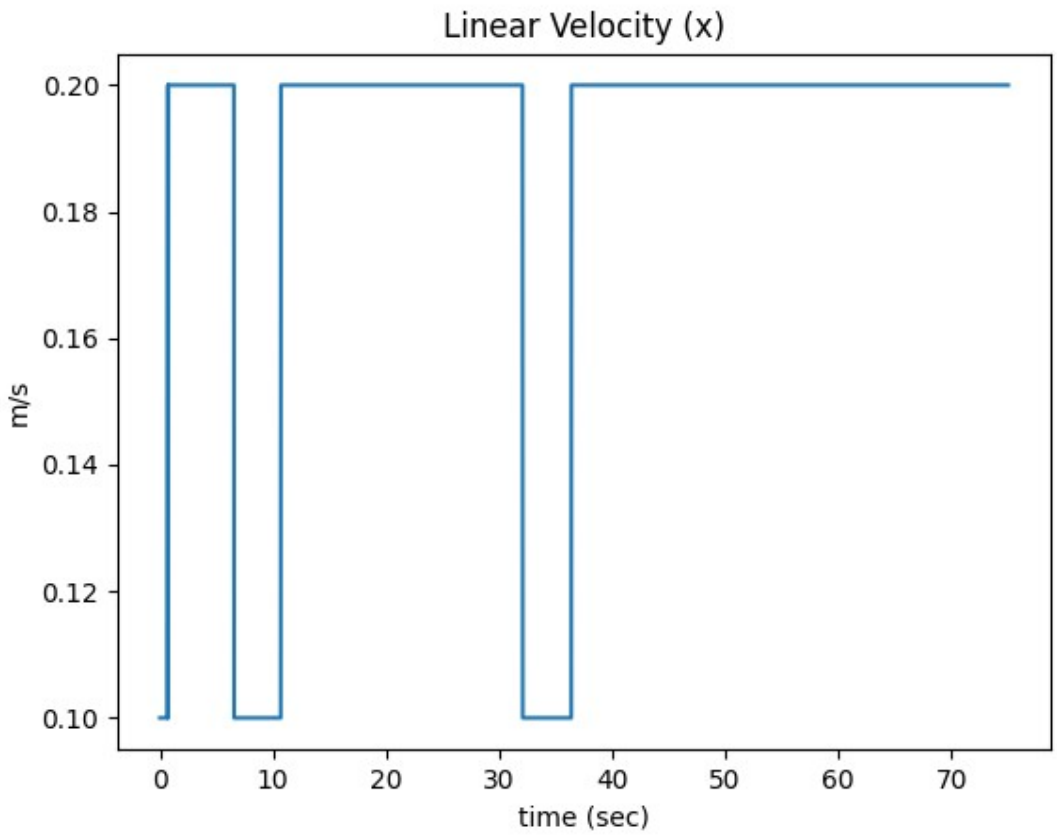
Τέλος, η σωστή λειτουργία του pd ελεγκτή φαίνεται στα διαστήματα μεταξύ των γωνιών (πχ 13-33sec) όπου η απόσταση είναι σταθερά ίση με **0.3m** (έχοντας φυσικα μικρομεταβολές που ο ελεγκτής διορθώνει άμεσα).

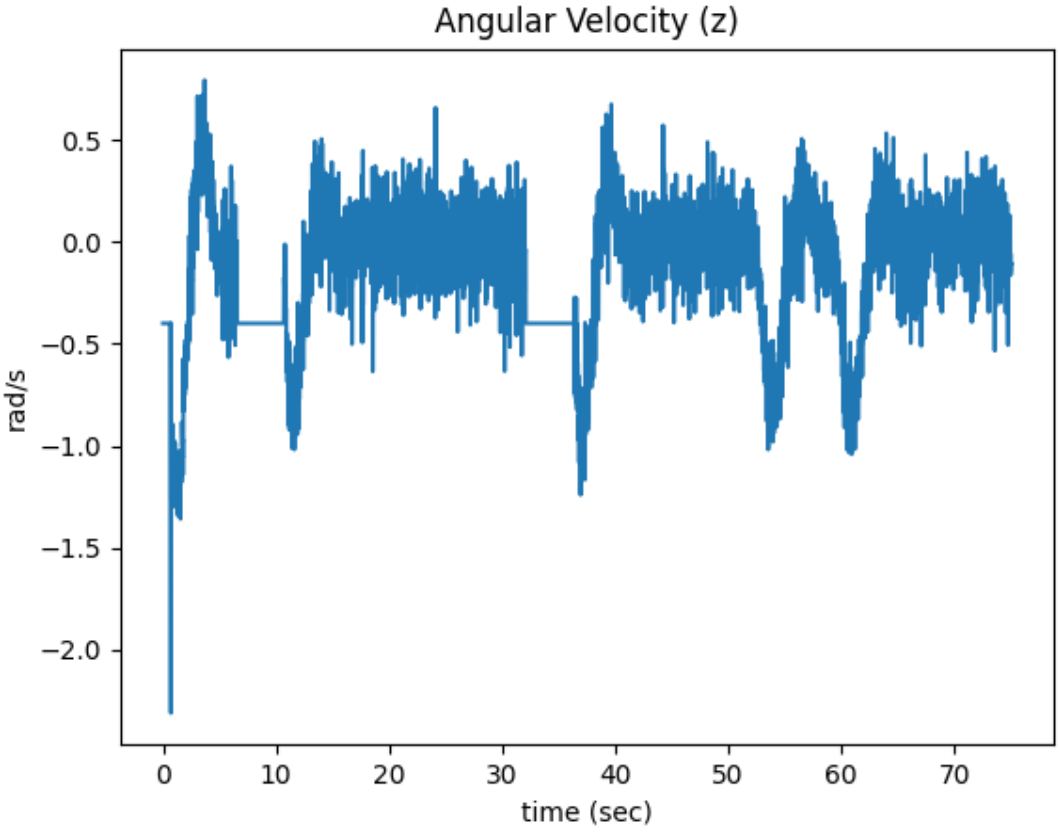


Στη διάγραμμα αυτό, φαίνεται η γωνία που σχηματίζει το ρομπότ με τον τοίχο, όπου όπως αναμέναμε, είναι ίση με μηδέν στις ευθείες, αφού το ρομπότ είναι παράλληλο στον τοίχο.

Ωστόσο, όπως αναφέραμε και πριν, υπάρχουν αναπόφευκτα οι μεγάλες μεταβολές, οι οποίες αντιστοιχούν στις γωνίες-στροφές και είναι πλήρως αναμενόμενες.

Τέλος, ακολουθούν οι γραφικές των ταχυτήτων (γραμμικής / γωνιακής), όπου η πρώτη δε μας δείχνει κάτι, αφού εμείς ορίζουμε τη γραμμική ταχύτητα στον x άξονα ανάλογα το state που βρίσκεται το ρομπότ, αλλά από τη δεύτερη βλέπουμε τη συνεχή προσπάθεια που κάνει ο pd ελεγκτής να διορθώσει το σφάλμα απόστασης, αυξομειώνοντας ανάλογα τη γωνιακή ταχύτητα στον άξονα z.


Linear Velocity (x)
0.20
0.18
0.16
0.14
0.12
0.10
m/s
0
10
20
30
40
50
60
70
time (sec)


Angular Velocity (z)
0.5
0.0
−0.5
−1.0
−1.5
−2.0
rad/s
0
10
20
30
40
50
60
70
time (sec)