Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Εξαμηνιαία Εργασία 2Α

Αυτοκινούμενα ρομπότ: Παρακολούθηση εμποδίου

Μάθημα: Ρομποτική ΙΙ: Ευφυή Ρομποτικά Συστήματα

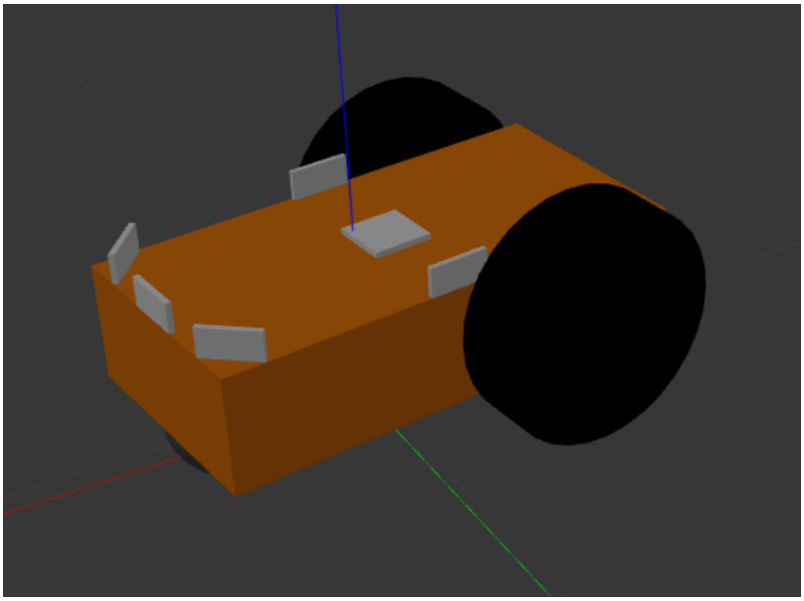
Εξάμηνο: 8ο

Ονοματεπώνυμα: Αλεξοπούλου Γεωργία, Γκενάκου Ζωή

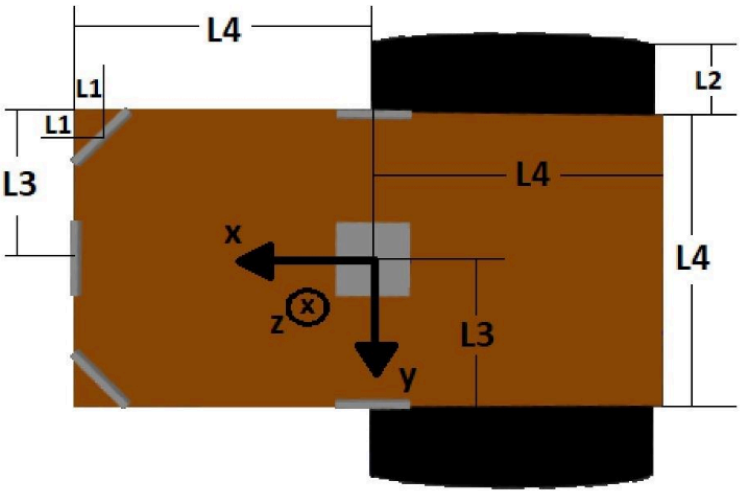
Εισαγωγή

Α. Το Ρομπότ

Σε αυτή την εργασία, καλούμαστε να αναπτύξουμε έναν αλγόριθμο για την παρακολούθηση ενός εμποδίου τοίχου (wall following). Για τον σκοπό αυτό χρησιμοποιήθηκε ένα ρομπότ διαφορικής οδήγησης, το οποίο έπρεπε να εκτελέσει τουλάχιστον μία πλήρη περιφορά γύρω από τα εμπόδια, διατηρώντας σταθερή απόσταση από αυτά.



Το ρομπότ που χρησιμοποιήθηκε είναι τύπου διαφορικής οδήγησης (differential drive) με δύο τροχούς διαμέτρου 20 cm, το οποίο δημιουργήθηκε σε περιβάλλον προσομοίωσης. Η συγκεκριμένη ρομποτική διάταξη, με διαστάσεις οι οποίες απεικονίζονται παρακάτω, σχεδιάστηκε και εξοπλίστηκε καταλλήλως για τους σκοπούς της παρούσας εργασίας.



$L_1 = 0.018\,m,\ L_2 = 0.05\,m,\ L_3 = 0.1\,m,\ L_4 = 0.2\,m$

Η ρομποτική διάταξη ενσωματώνει:

1) 5 αισθητήρες υπερήχων σόναρ για τη μέτρηση αποστάσεων από εμπόδια
2) Ένα Inertial Measurement Unit (IMU) 9 βαθμών ελευθερίας, που μετράει στροφικές ταχύτητες, επιταχύνσεις και περιστροφή γύρω από κάθε άξονα.

Β. Θεωρητική Ανάλυση

Λαμβάνοντας υπόψη τους αριθμούς μητρώου μας, ορίζουμε τις μεταβλητές $X_1 = 4,\ X_2 = 5$ για τον υπολογισμό του αρχικού προσανατολισμού της ρομποτικής διάταξης, έχουμε την εξής γωνία:

$$ang = mod(X_1 + X_2, \pi) = mod(9, \pi) = 2.71681469282$$

Επιπλέον, καθώς το άθροισμα των μεταβλητών $X_1,\ X_2$ είναι ίσο με $X_1 + X_2 = 9$, δηλαδή περιττός, η φορά της κίνησης που θα ακολουθήσει το ρομπότ είναι αντιωρολογιακή (CCW).

Ο αλγόριθμος που έχουμε κατασκευάσει αναγνωρίζει 3 διεργασίες κατά τη διάρκεια της κίνησης: αρχική τοποθέτηση στο πλησιέστερο εμπόδιο, ευθυγράμμιση του ρομπότ με το εμπόδιο και κίνηση παράλληλη ως προς το εμπόδιο.

Ο αλγόριθμος καλείται, αρχικά, να τοποθετήσει το ρομπότ κοντά στο εμπόδιο (τοίχος). Για τον σκοπό αυτό, εκτελείται κίνηση με χαρακτηριστικά:

`self.velocity.linear.x = 0.6` και `self.velocity.angular.z = 0.0`

Το ρομπότ κινεί ευθύγραμμη ομαλή κίνηση κατά τον άξονα x. Όταν το εμπόδιο να απέχει λιγότερο από 0.4 m από τον μπροστινό αισθητήρα,

`self.sonar_R.range, self.sonar_FR.range`

Τότε ο αλγόριθμος μεταβαίνει στη δεύτερη διεργασία ελέγχου κίνησης του ρομπότ.

Κατά τη διάρκεια της διαδικασίας της ευθυγράμμισης, το ρομπότ εκτελεί στροφική κίνηση, ώστε να τοποθετηθεί παράλληλα με το εμπόδιο. Ο αλγόριθμος οδηγείται σε αυτή τη διεργασία είτε μετά την τοποθέτηση του ρομπότ κοντά στο εμπόδιο (όπως αναλύθηκε παραπάνω), είτε μετά την εκτέλεση παράλληλης κίνησης με το εμπόδιο, όταν τελικά βρεθεί σε γωνία. Συνεπώς, υπεύθυνος για τη μετάβαση μεταξύ των διεργασιών 1-2 και 3-2 είναι ο εμπρόσθιος αισθητήρας το ρομπότ, ο οποίος ελέγχει την απόστασή του από τον τοίχο. Για την εκτέλεση της περιστροφής, ωστόσο, δεν αρκεί μόνο ο έλεγχος από τον `self.sonar_F.range` αισθητήρα˙ αντιθέτως, είναι απαραίτητος και ο έλεγχος από την πλευρά του εμποδίου. Έτσι, και καθώς η φορά περιστροφής είναι αντιωρολογιακή, ελέγχονται οι αποστάσεις των αισθητήρων στη δεξιά πλευρά του ρομπότ (`self.sonar_R.range, self.sonar_FR.range`).

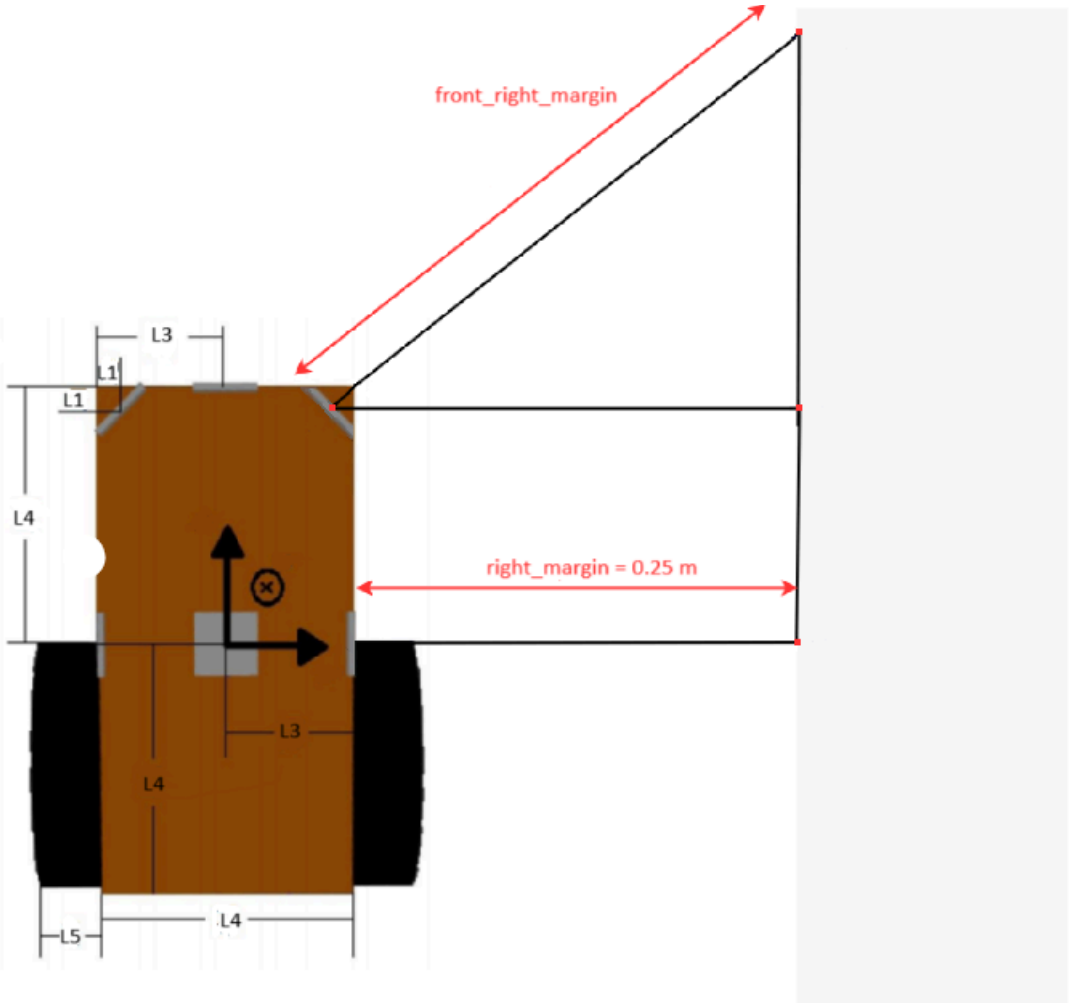
Για την περιστροφή, δίνονται τα μεγέθη `self.velocity.linear.x = 0.025` και `self.velocity.angular.z = -np.pi`. Καθώς η επιθυμητή κίνηση είναι κυρίως στροφική, δίνουμε στη γωνιακή ταχύτητα μια αρκετά μεγάλη τιμή σε σχέση με προηγουμένως (αρνητική για να εξυπηρετεί την CCW κίνηση) και μια πολύ μικρή, αλλά όχι μηδενική, τιμή στη γραμμική ταχύτητα κατά μήκος του άξονα x. Με αυτόν τον τρόπο, κατά την περιστροφή δεν σταματάει η γραμμική μετακίνηση του ρομπότ. Ωστόσο, η γραμμική ταχύτητα που έχει επιλεχθεί είναι πολύ μικρή, ώστε να αποφευχθεί η σύγκρουση του ρομπότ με το εμπόδιο. Η διεργασία αυτή ολοκληρώνεται όταν η απόσταση των αισθητήρων στη δεξιά πλευρά γίνεται μικρότερη από την απόσταση του μπροστινού αισθητήρα από το εμπόδιο:

```
(self.sonar_R.range + 0.25 < self.sonar_F.range) and
 (self.sonar_FR.range + 0.25 < self.sonar_F.range)
```

Για τη διασφάλιση της σωστής αρχικοποίησης του ρομπότ παράλληλα με το εμπόδιο, προστίθεται ο όρος `0.25`, ο οποίος ελαττώνει την ανάγκη επανάληψης του βήματος ευθυγράμμισης κατά τη διάρκεια της κίνησης.

Με την ολοκλήρωση της διαδικασίας ευθυγράμμισης του ρομπότ με το εμπόδιο, το ρομπότ εκτελεί παράλληλη κίνηση με το αυτό. Για τον σκοπό αυτό, κινείται με σταθερή γραμμική ταχύτητα `self.velocity.linear.x = 0.6` και γωνιακή ταχύτητα `self.velocity.angular.z = -(Kp * proportional_error + Kd * derivative_error)`, έτσι ώστε η απόσταση του ρομπότ από το εμπόδιο να παραμένει σταθερή έως ότου βρεθεί και πάλι σε γωνία (όπου θα χρειαστεί να εκτελέσει εκ νέου τη διαδικασία της ευθυγράμμισης). Παρατηρούμε πως η γωνία στροφής ορίζεται μέσω του PID controller και είναι αρνητική, προκειμένου η κίνηση που εκτελείται να έχει την επιθυμητή αντιωρολογιακή φορά. Καθ' όλη τη διάρκεια της κίνησης, η απόσταση από το εμπόδιο ελέγχεται με την αξιοποίηση των μετρήσεων των δεξιών αισθητήρων του ρομπότ (`sonar_R, sonar_FR`), οι οποίες αξιοποιούνται στην ελαχιστοποίηση των σφαλμάτων τους από τις αναμενόμενες τιμές τους.

Συγκεκριμένα, η σταθερή απόσταση που επιθυμούμε να κρατά το ρομπότ στα δεξιά του από τον τοίχο είναι συνεχώς ίση με `right_margin = 0.25`, ενώ η τιμή που πρέπει να διατηρεί ο εμπρόσθιος-δεξιά αισθητήρας δίνεται από τη σχέση:



$$\frac{right\_margin + L1}{front\_right\_margin} = \frac{\sqrt{2}}{2}$$ (καθώς το τρίγωνο που σχηματίζεται είναι ισοσκελές)

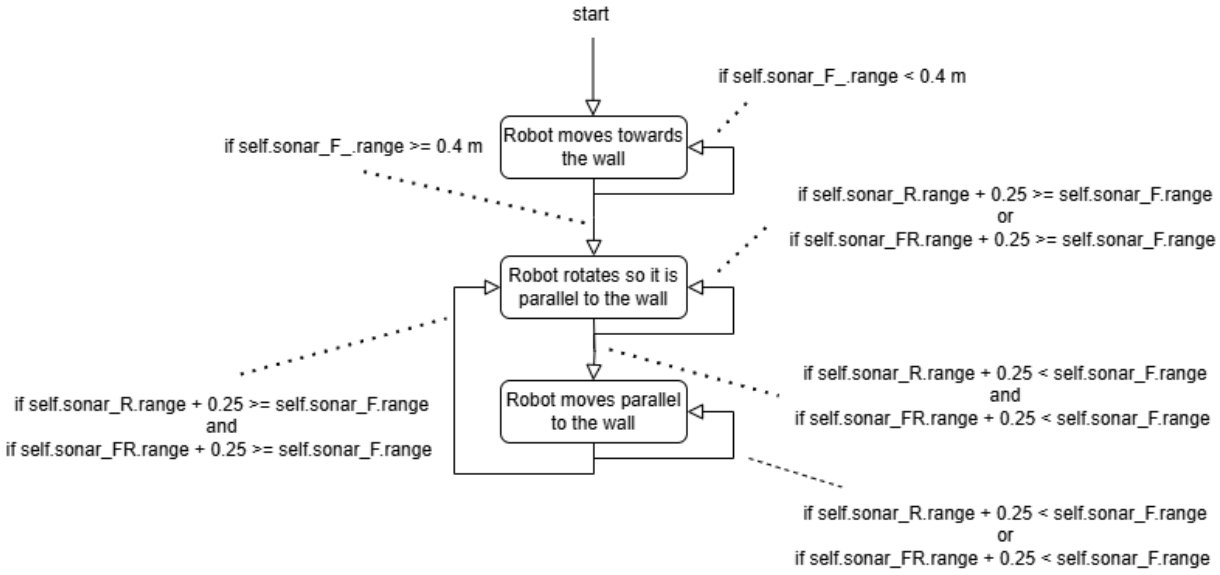
$$\Leftrightarrow \frac{0.25 + 0.018}{front\_right\_margin} = \frac{\sqrt{2}}{2} \Leftrightarrow \frac{0.268}{front\_right\_margin} = \frac{\sqrt{2}}{2} \Leftrightarrow front\_right\_margin \simeq 0.38\, m$$

Αξίζει να σχολιαστεί πως οι τιμές των σφαλμάτων του ελεγκτή υπολογίζονται ως εξής:

- `proportional_error = front_right_error + right_error`, όπου
  `front_right_error = front_right_margin - self.sonar_FR.range` και
  `right_error = right_margin - self.sonar_R.range`

- `derivative_error = (derivative_front_right + derivative_right) / dt`, όπου
  `derivative_right = right_error - previous_P_R_error` και
  `derivative_front_right = front_right_error - previous_P_FR_error`

Ο αλγόριθμος, όπως τον περιγράψαμε, μπορεί να αναπαρασταθεί από το παρακάτω flow chart:

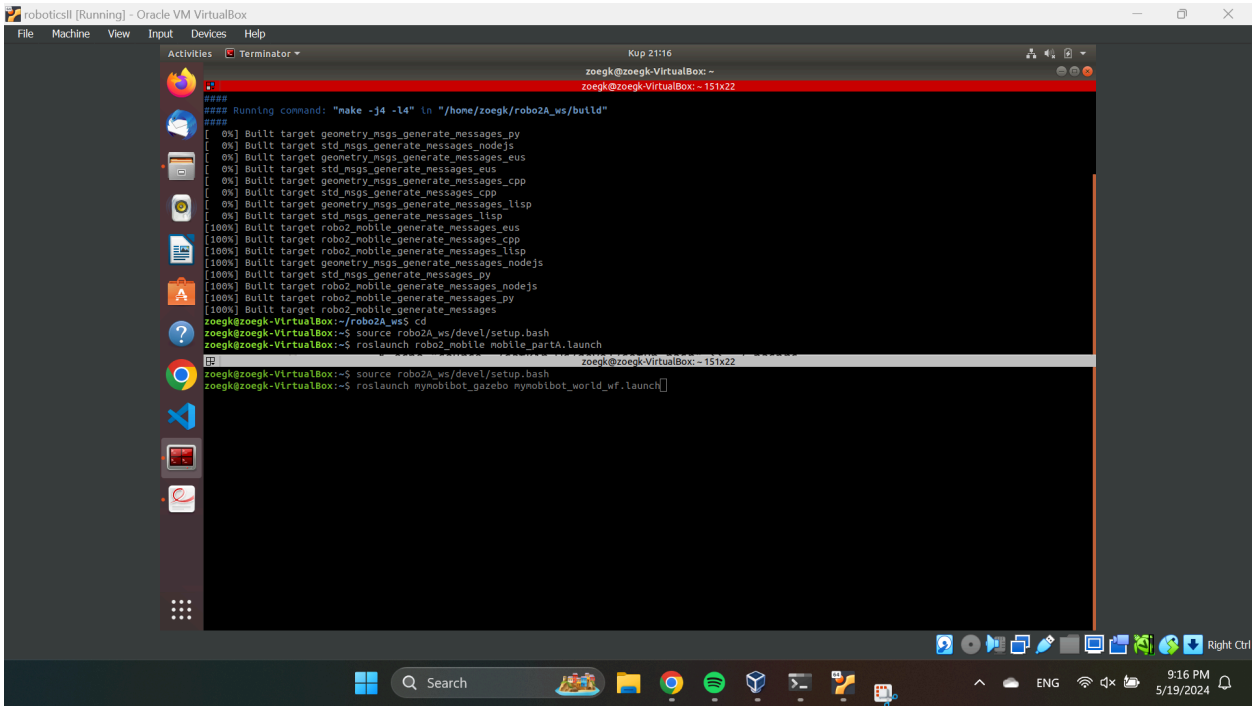


C. Προσομοίωση

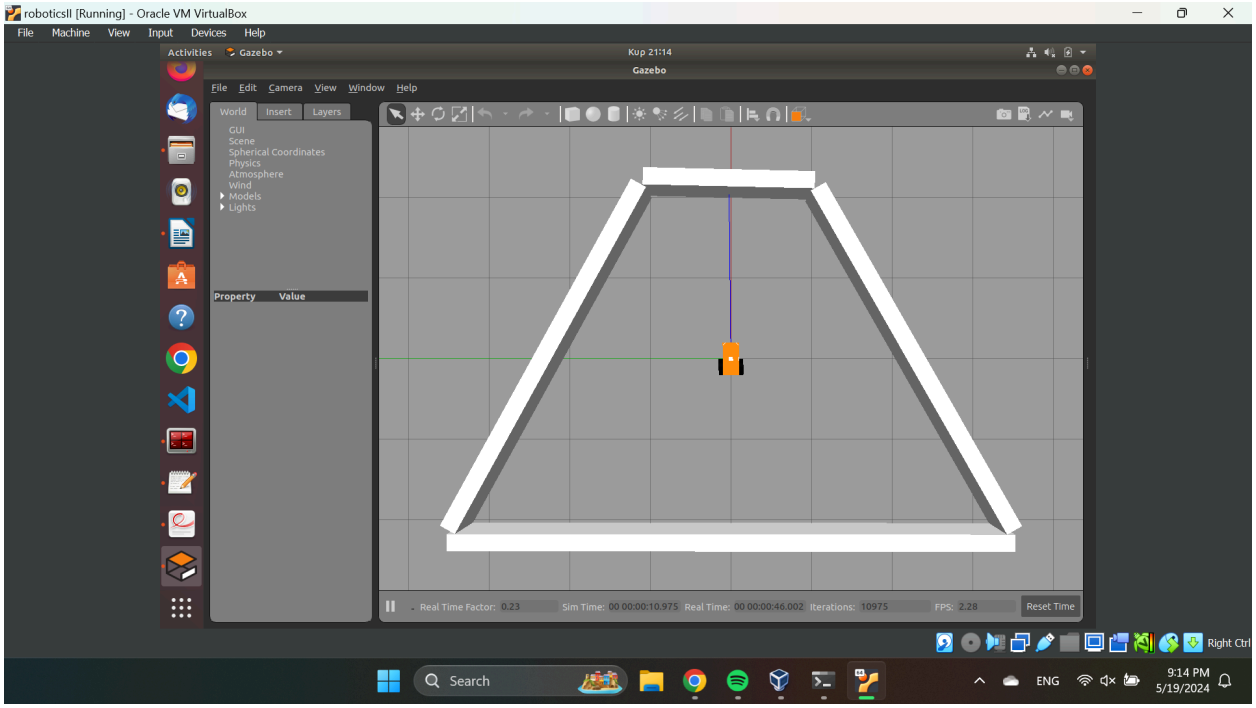
Για το περιβάλλον προσομοίωσης της εργασίας, εργαστήκαμε με τα εργαλεία ROS Melodic και Gazebo.
Αρχικά, δημιουργήσαμε ένα περιβάλλον εργασίας με το όνομα `robo2A_ws`, και στη διεύθυνση `~/robo2A_ws/src` εγκαθιστούμε το τα πακέτα που αναφέρονται στις οδηγίες (ros-control, ros-controllers, ros-navigation και gazebo_ros_pkgs), τον φάκελο robo2_mobile που μας δίνεται και κλωνοποιούμε το github directory με link https://github.com/oikonpar/mymobibot.git .
Σύμφωνα με την παραπάνω θεωρητική ανάλυση, κατασκευάζουμε τον κώδικα για το follower.py.
Για την εκτέλεση του ros πακέτου μας, οφείλουμε να το αρχικοποιούμε με την παρακάτω εντολή, κάθε φορά που ανοίγουμε καινούριο terminal:

```
source robo2A_ws/devel/setup.bash
```

Για την εκτέλεση του project, θα ανοίξουμε 2 terminals. Το ένα για να ανοίξουμε το Gazebo και ένα για να ξεκινήσει ο κώδικας να εκτελείται.

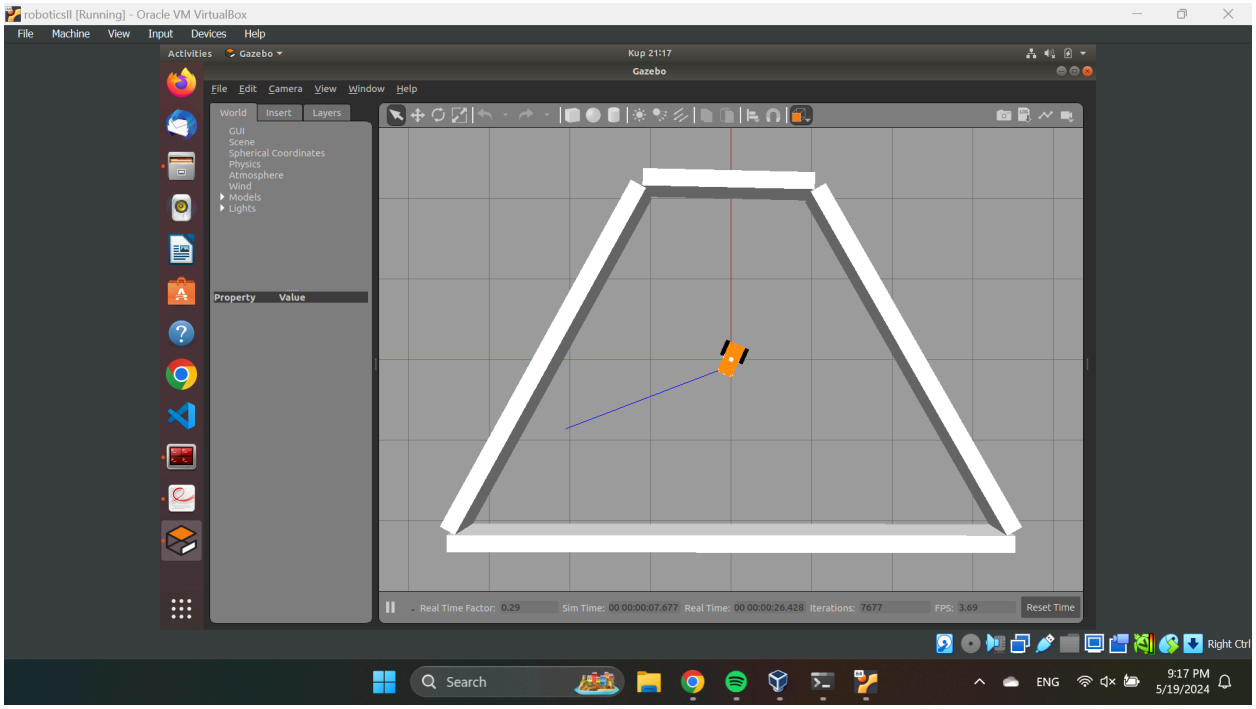


Η εκτέλεση της εντολής `roslaunch mymobibot_gazebo mymobibot_world_wf.launch` εκκινεί το πρόγραμμα Gazebo, στο οποίο υπάρχει το mobile robot με τα εμπόδια που το περιτριγυρίζουν.

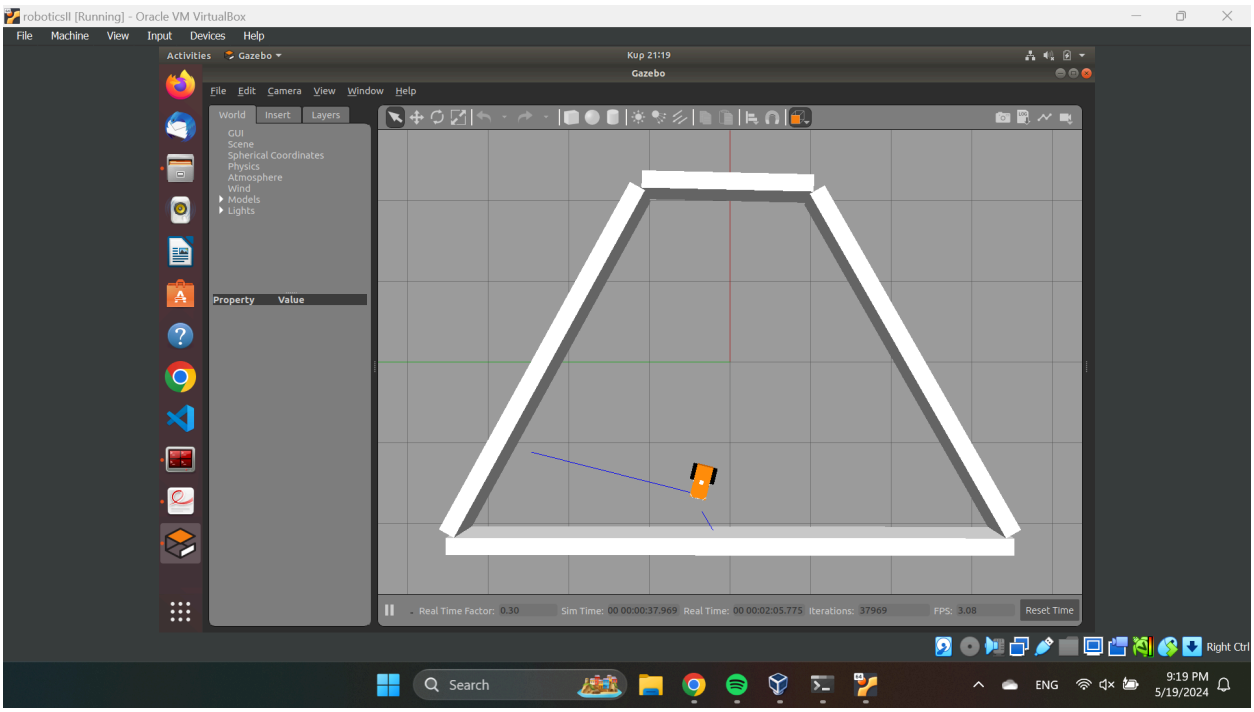


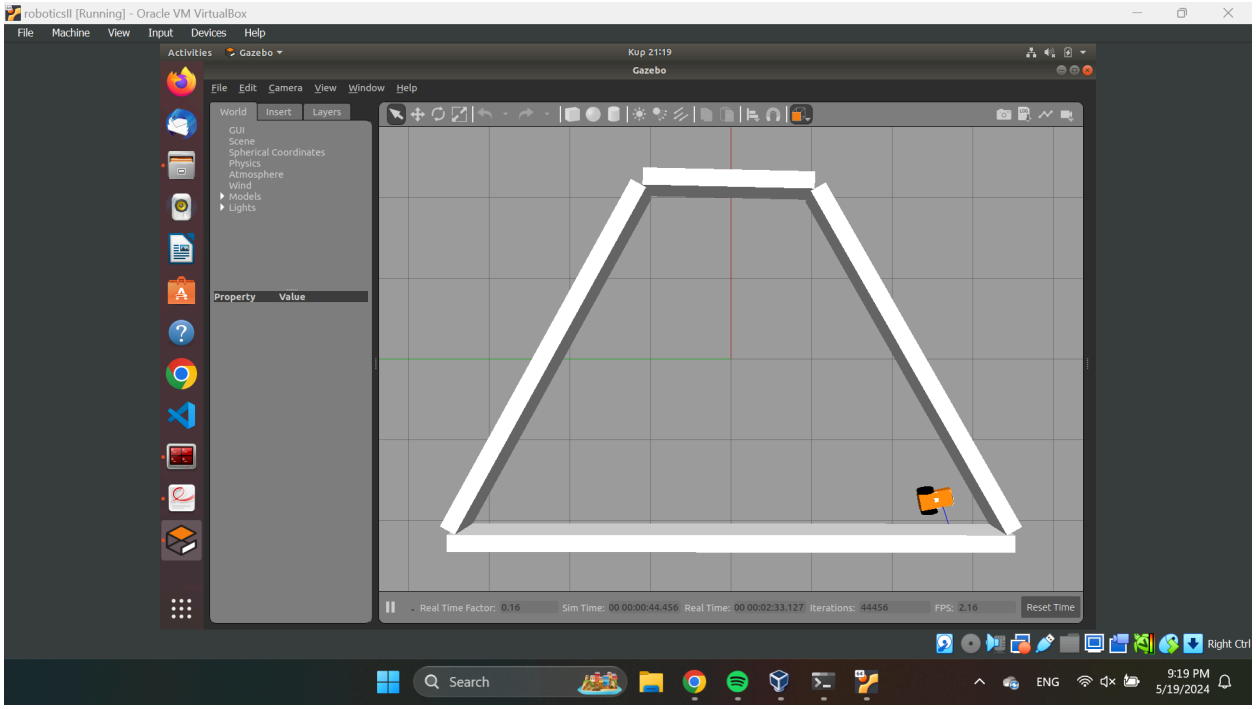
Όπως υποδεικνύει και η εκφώνηση της άσκησης μεταβάλλουμε κάποιες παραμέτρους αρχικοποίησης, βάσει του τελευταίου ψηφίου του αριθμού μητρώου μας. Ο κωδικός αριθμός της ομάδας μας είναι X = 5+4 = 9.

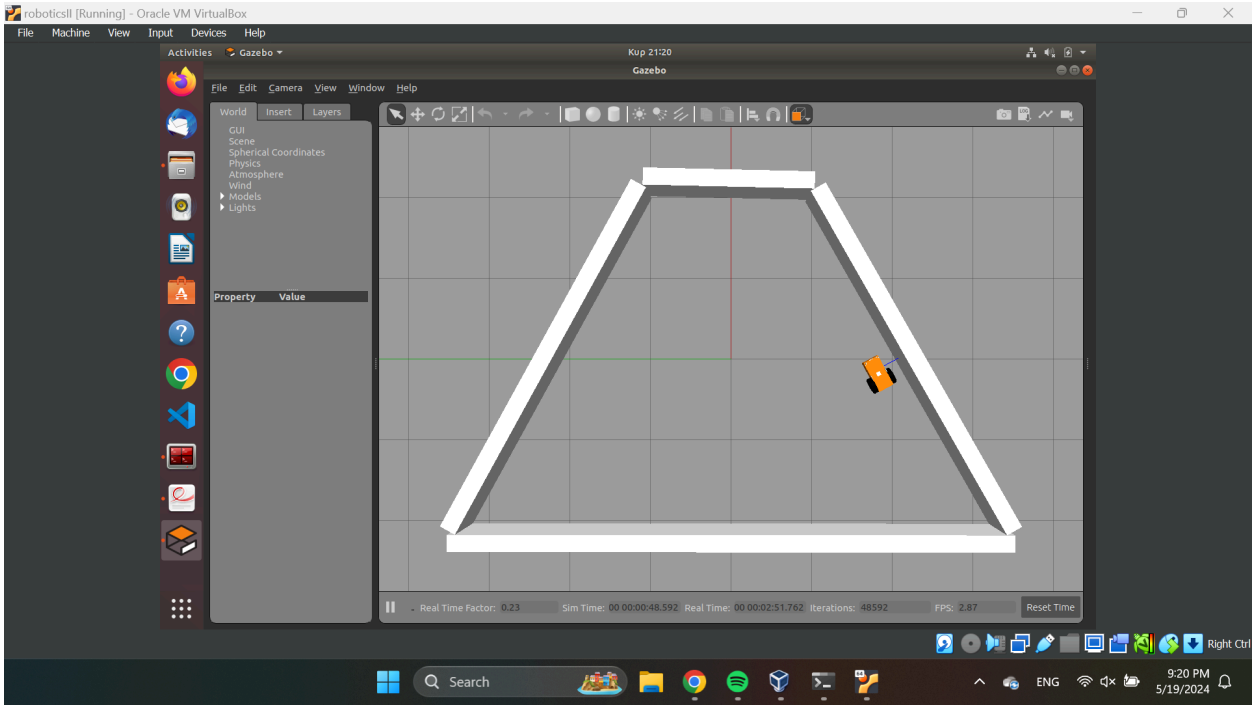
Με βάση αυτό, θέτουμε τον αρχικό προσανατολισμό του ρομπότ σε γωνία: angle = mod(X,π) = 2.71681469282. Για την μεταβολή του αρχικού προσανατολισμού (ως προς τον άξονα z) του ρομπότ, ανοίγουμε το αρχείο mymobibot_world_wf.launch στον φάκελο `~/catkin_ws/src/mymobibot/mymobibot_gazebo/launch/` και θέτουμε την τιμή *angle*: `<arg name="yaw_init" default="2.71681469282"`. Επίσης, επειδή ο κωδικός αριθμός μας είναι περιττός, η παρακολούθηση του εμποδίου γίνεται με αντι-ωρολογιακή φορά ως προς το παγκόσμιο σύστημα συντεταγμένων.
Κάνουμε την αλλαγή και βλέπουμε όντως την διαφορά στην γωνία αρχικοποίησης:

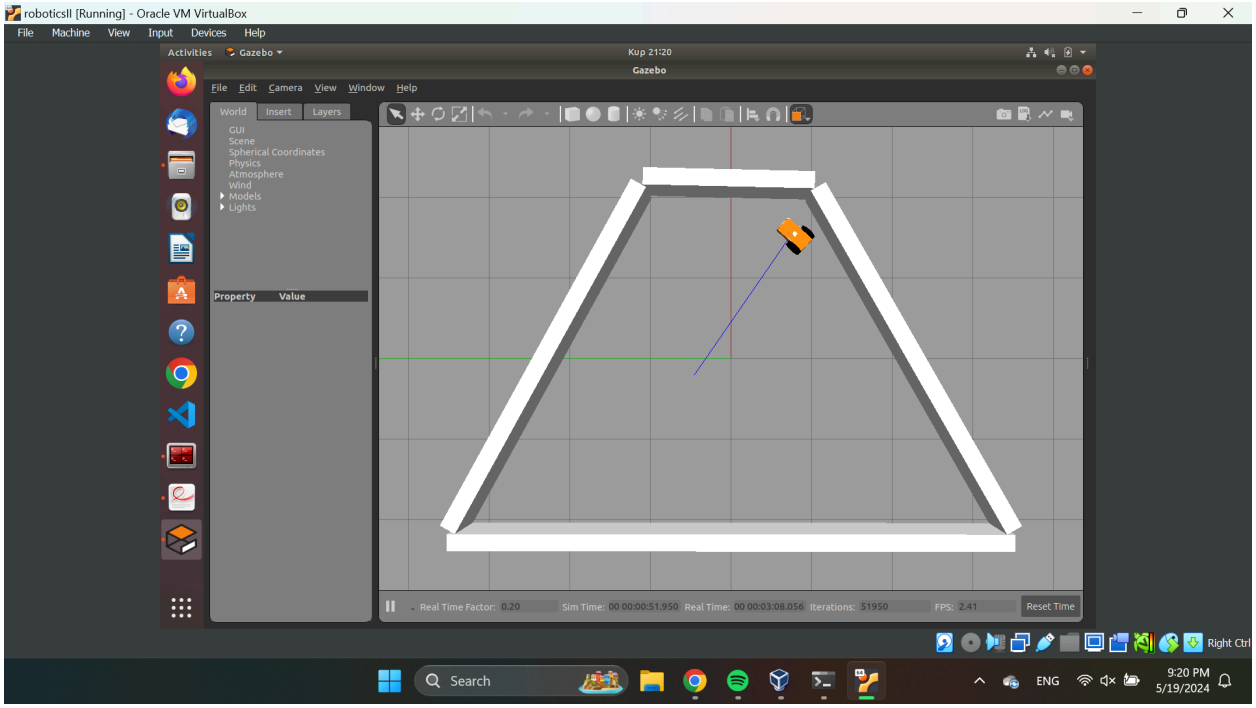


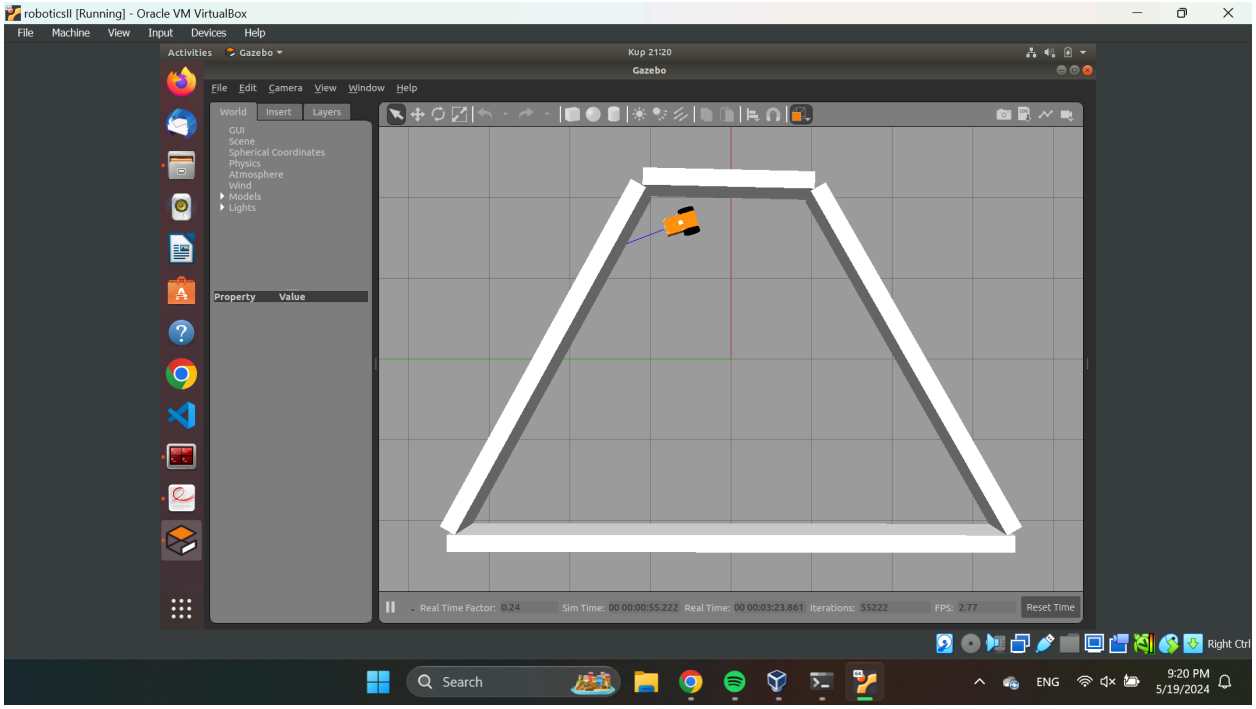
Εκτελώντας την εντολή `roslaunch robo2_mobile mobile_partA.launch`, το ρομπότ αρχίζει να κινείται, εκτελώντας κίνηση μεταξύ των εμποδίων. Στις παρακάτω εικόνες φαίνεται η πορεία που ακολουθεί από την θέση αρχικοποίησης εώς ότου κάνει μια πλήρη περιστροφή.

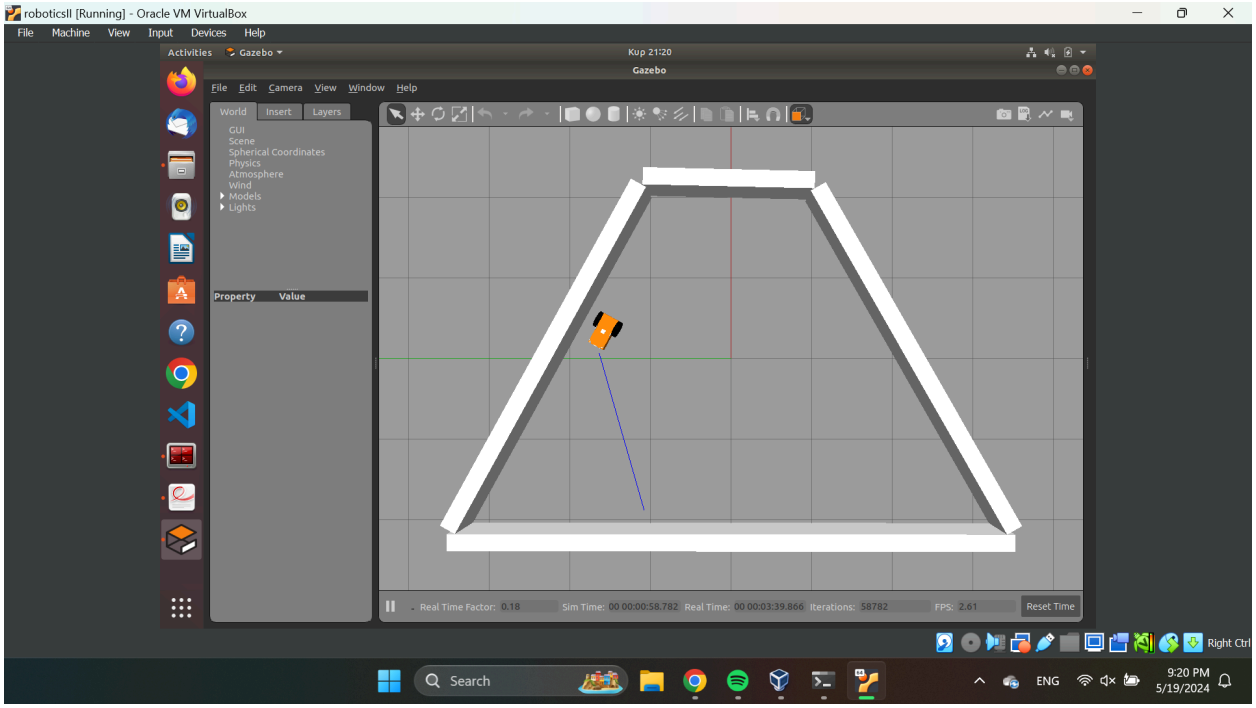


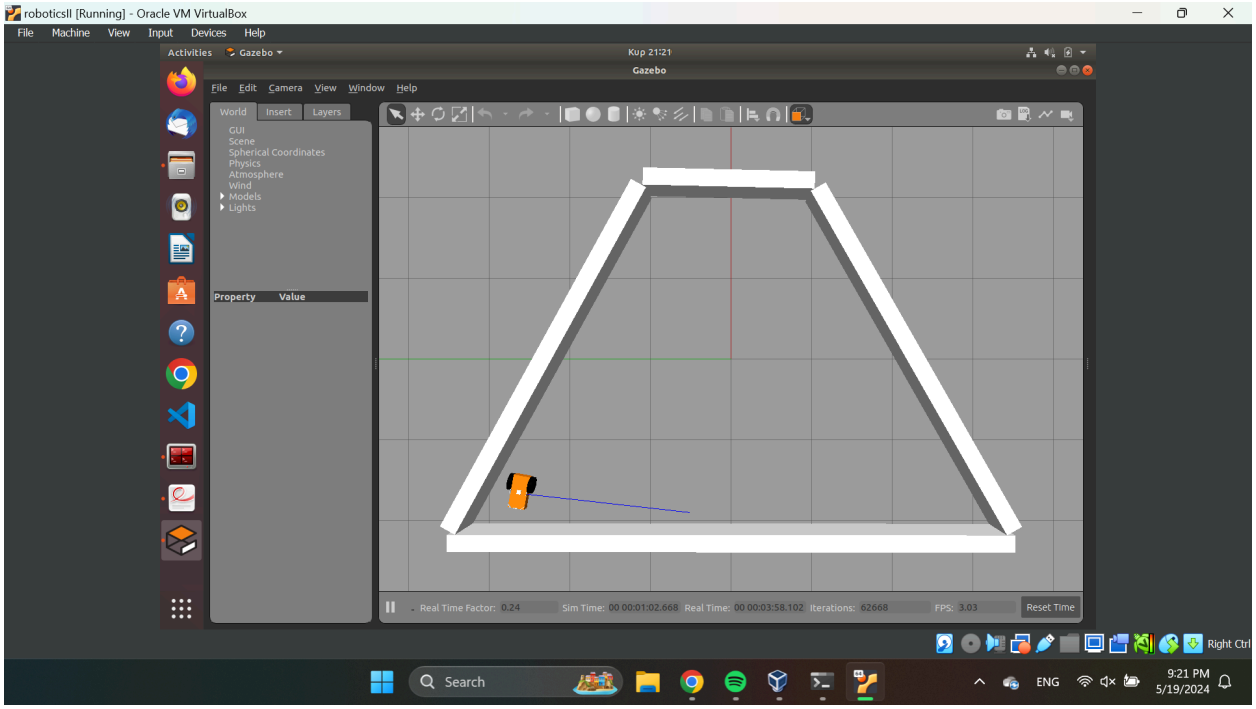

roboticsII [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Gazebo Kup 21:19
Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
Real Time Factor: 0.16 Sim Time: 00 00:00:44.456 Real Time: 00 00:02:33.127 Iterations: 44456 FPS: 2.16 Reset Time
Right Ctrl
Search
ENG
9:19 PM
5/19/2024

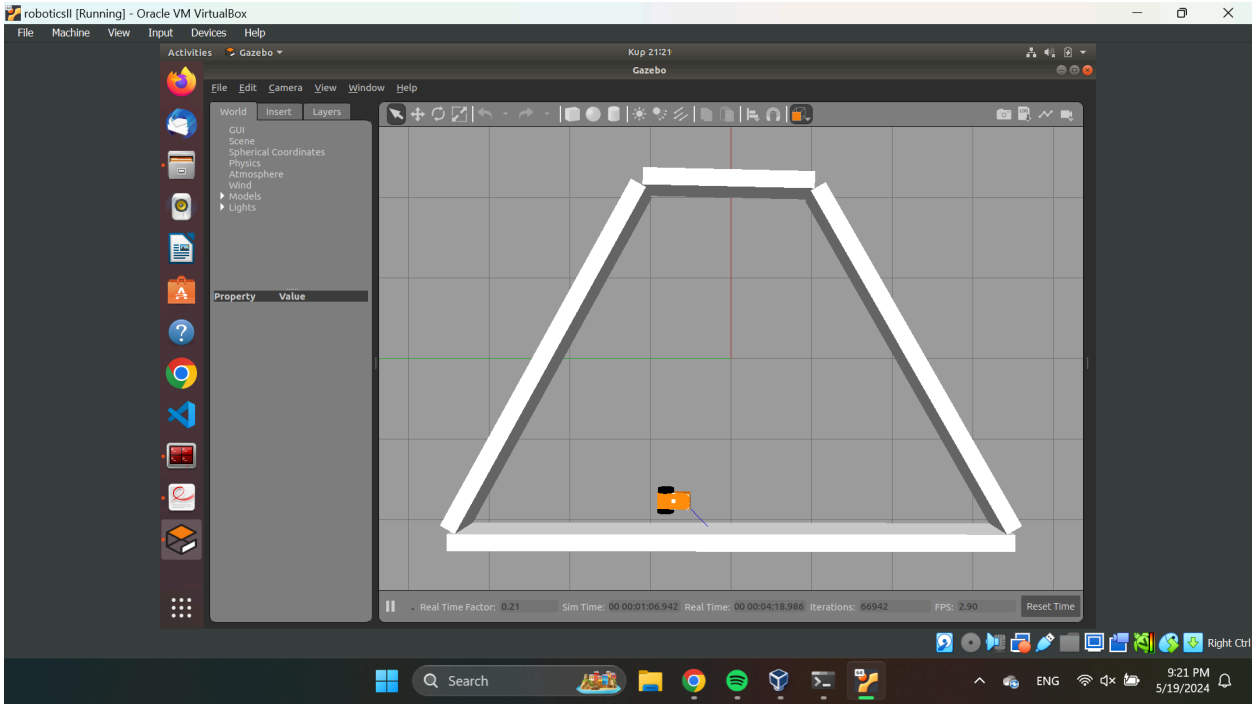

roboticsII [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Gazebo Kup 21:20
Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
Real Time Factor: 0.23 Sim Time: 00 00:00:48.592 Real Time: 00 00:02:51.762 Iterations: 48592 FPS: 2.87 Reset Time
Right Ctrl
Search
ENG
9:20 PM
5/19/2024

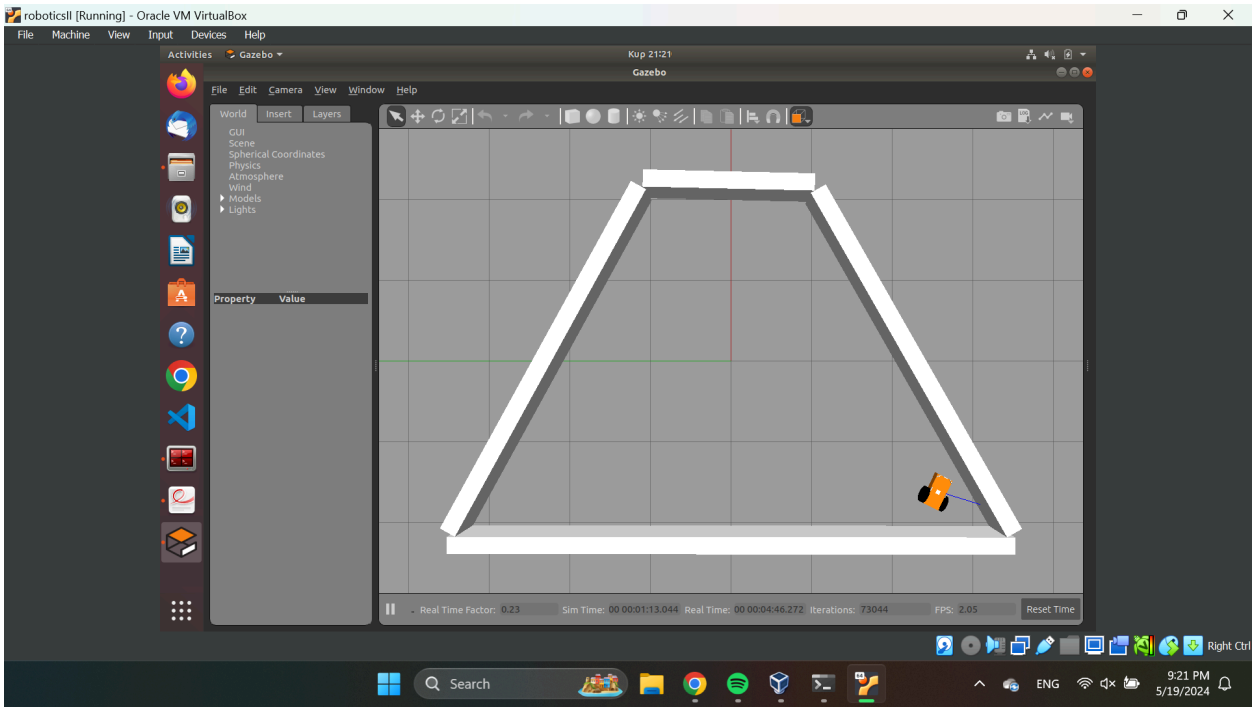

roboticsII [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Gazebo Kup 21:20
Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
Real Time Factor: 0.20 Sim Time: 00 00:00:51.950 Real Time: 00 00:03:08.056 Iterations: 51950 FPS: 2.41 Reset Time
Right Ctrl
Search
ENG
9:20 PM
5/19/2024


roboticsII [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Gazebo Kup 21:20
Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
Real Time Factor: 0.24 Sim Time: 00 00:00:55.222 Real Time: 00 00:03:23.861 Iterations: 55222 FPS: 2.77 Reset Time
Right Ctrl
Search
ENG
9:20 PM
5/19/2024


roboticsII [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Gazebo Kup 21:20
Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
Real Time Factor: 0.18 Sim Time: 00 00:00:58.782 Real Time: 00 00:03:39.866 Iterations: 58782 FPS: 2.61 Reset Time
Right Ctrl
Search
ENG
9:20 PM
5/19/2024


roboticsII [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Gazebo Kup 21:21
Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
Real Time Factor: 0.24 Sim Time: 00 00:01:02.668 Real Time: 00 00:03:58.102 Iterations: 62668 FPS: 3.03 Reset Time
Right Ctrl
Search
ENG
9:21 PM
5/19/2024



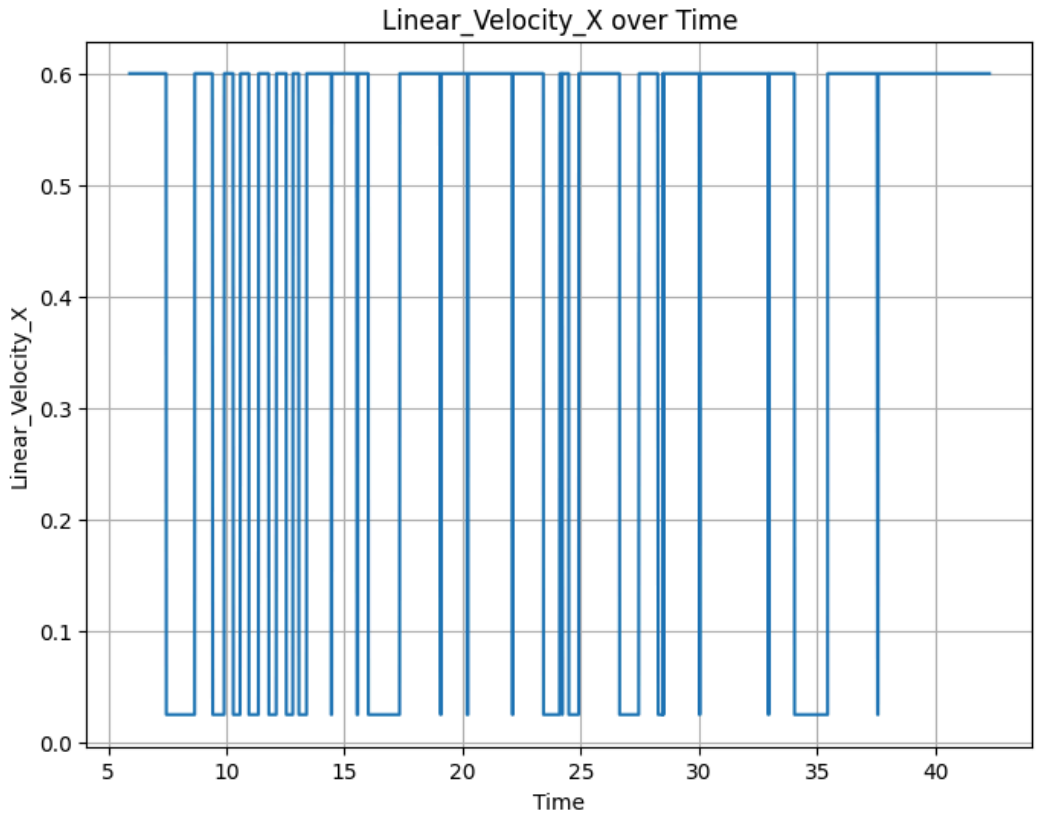


Συγκεκριμένα, το ρομπότ κινείται με ταχύτητα 0.6m/s μέχρι να συναντήσει εμπόδιο. Στην επόμενη εικόνα συναντά εμπόδιο και στρίβει CCW. Στις επόμενες εικόνες φαίνεται το ρομπότ να ακολουθεί τα εμπόδια με βοήθεια του PD ελεγκτή και συνεχίζει να ακολουθεί τα εμπόδια μέχρι να σταματήσουμε manually την προσομοίωση.

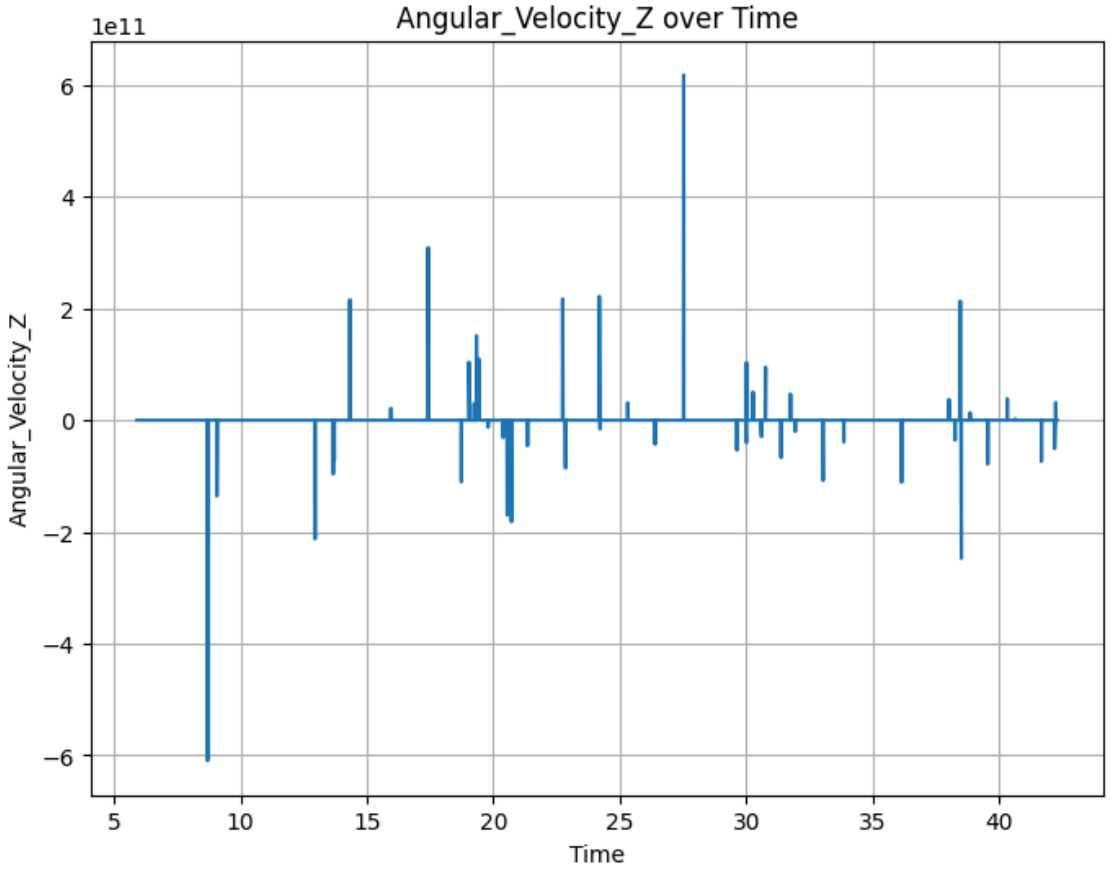
Για μια πλήρη περιστροφή, αποθηκεύουμε δεδομένα για τις ταχύτητες και τα σφάλματα. Χρησιμοποιούμε τον παρακάτω κώδικα για να δημιουργήσουμε τα επόμενα διαγράμματα.

```
import matplotlib.pyplot as plt

# Read data from the file
data = {'Time': [], 'Linear_Velocity_X': [], 'Angular_Velocity_Z':
[],
        'Front_Right_Error': [], 'Right_Error': []}

with open('output.txt', 'r') as file:
    lines = file.readlines()
    headers = lines[0].strip().split(', ')
    for line in lines[1:]:
        values = line.strip().split(', ')
        for i, header in enumerate(headers):
            data[header].append(float(values[i]))

# Plot data
for header in headers[1:]:
    plt.figure(figsize=(8, 6))
    plt.plot(data['Time'], data[header])
    plt.xlabel('Time')
    plt.ylabel(header)
    plt.title(header + ' over Time')
    plt.grid(True)
    plt.show()
```

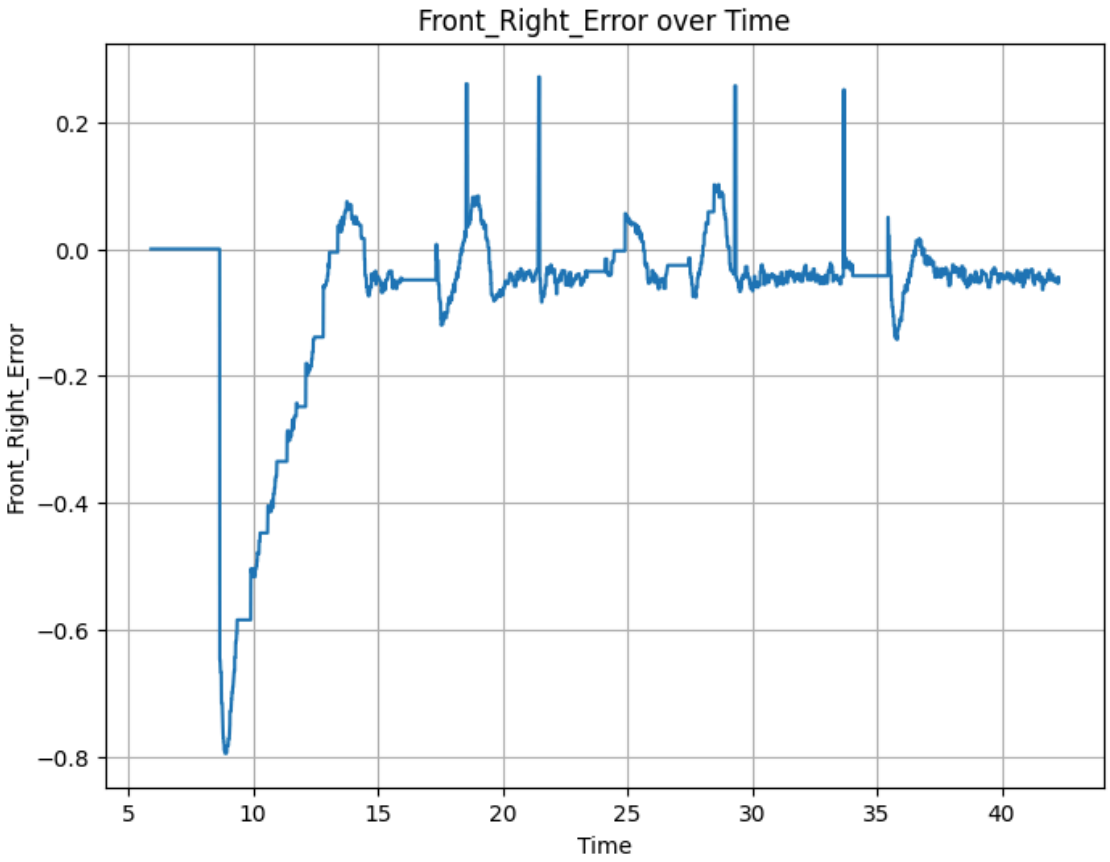
Στην παρακάτω εικόνα, παρατίθεται η γραμμική ταχύτητα του οχήματος σε σχέση με τον χρόνο. Η γραμμική ταχύτητα παίρνει τιμές 0.6m/s ή 0.025m/s, όπως έχουμε ορίσει άλλωστε στον κώδικα. Για το διάστημα το οποίο η ταχύτητα είναι σταθερή και ίση με 0.6m/s (ακόμα και με μεμονωμένα spikes), πρόκειται για απλή ακολούθηση εμποδίων. Ενώ, όταν η ταχύτητα είναι σταθερά 0.025m/s, το ρομπότ εκτελεί την υποεργασία στροφής. Τα μεμονωμένα spikes που παρατηρούνται προέρχονται από τον έλεγχο που μπορεί να γίνεται.

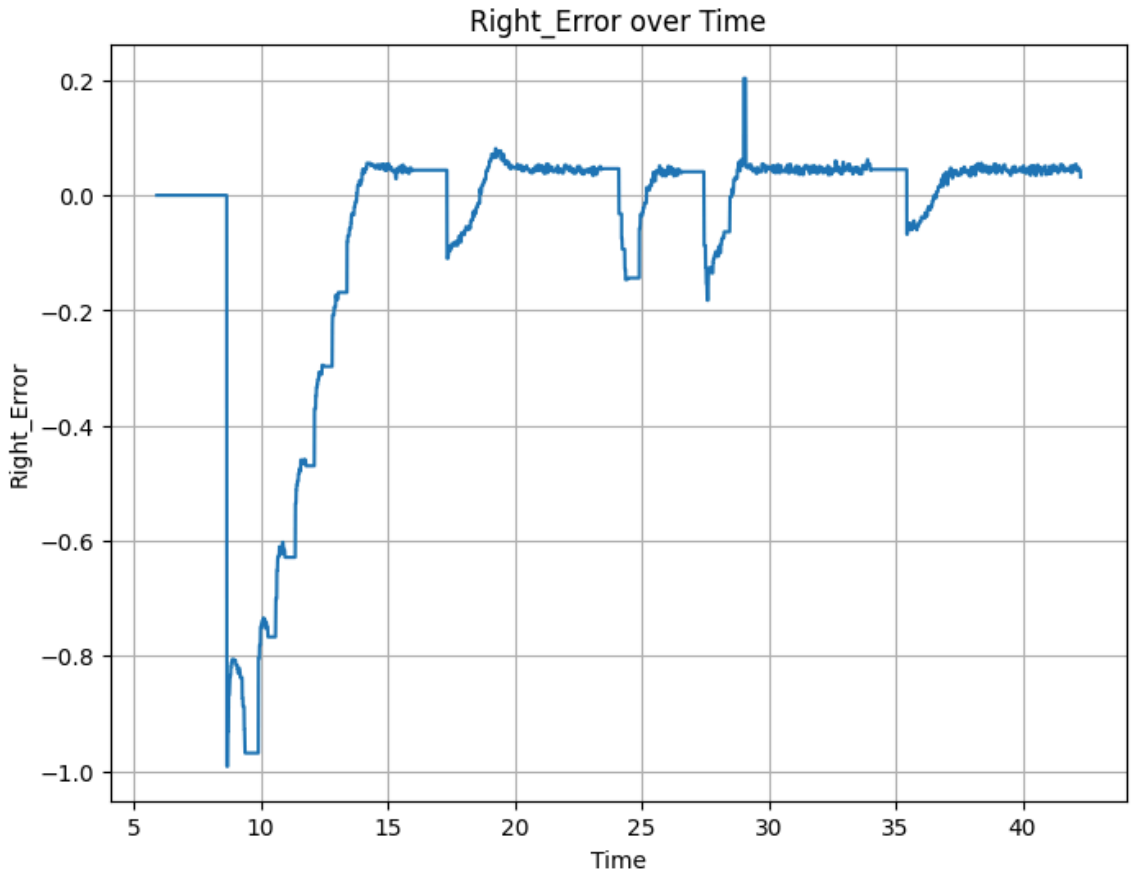


Στην παρακάτω εικόνα, παρατηρούμε την γωνιακή ταχύτητα του οχήματος συναρτήσει του χρόνου. Παρατηρούμε πως η γωνιακή ταχύτητα παρουσιάζει αρκετές διακυμάνσεις. Οι διακυμάνσεις αυτές οφείλονται στην επενέργηση του ελεγκτή.



Επιπλέον, επιθυμούμε να ελέγξουμε πόσο καλά εκτελέι τις επιθυμητές εργασίες του ρομπότ. Αυτό μπορούμε να το κάνουμε μελετώντας τα σφάματα των αισθητήρων που χρησιμοποιήσαμε. Παρακάτω, θα δούμε τα σφάλματα του δεξιού και του μπροστά δεξιού αισθητήρα. Στον κώδικα ορίζουμε την επιθυμητή απόσταση που θέλουμε να έχουμε από τον τοίχο μείων την απόσταση που μετράει ο αισθητήρας. Παρατηρούμε ότι αρχικά ότι το σφάλμα είναι μεγάλο (κατά απόλυτη τιμή), καθώς το ρομπότ κινείται από την θέση αρχικοποίησης προς τον τοίχο. Όσο το ρομπότ κινείται παράλληλα με τον τοίχο το σφάλμα φαίνεται να είναι πολύ κοντά στο μηδέν, ενώ μπορούμε να παρατηρήσουμε και peaks στα διαγράμματα των errors που αντιστοιχούν στις καταστάσεις που το ρομπότ πλησιάζει σε γωνίες και αναμενόμενα το σφάλμα θα παρουσιάσει απότομη αύξηση, αφού τότε ενεργοποιείται η υποεργασία στροφής.




Right_Error over Time
Right_Error
Time
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
5
10
15
20
25
30
35
40