

به نام حضرت دوست



دانشگاه امیرکبیر  
دانشکده مهندسی کامپیوتر

---

رباطیک  
تمرین سری اول

---

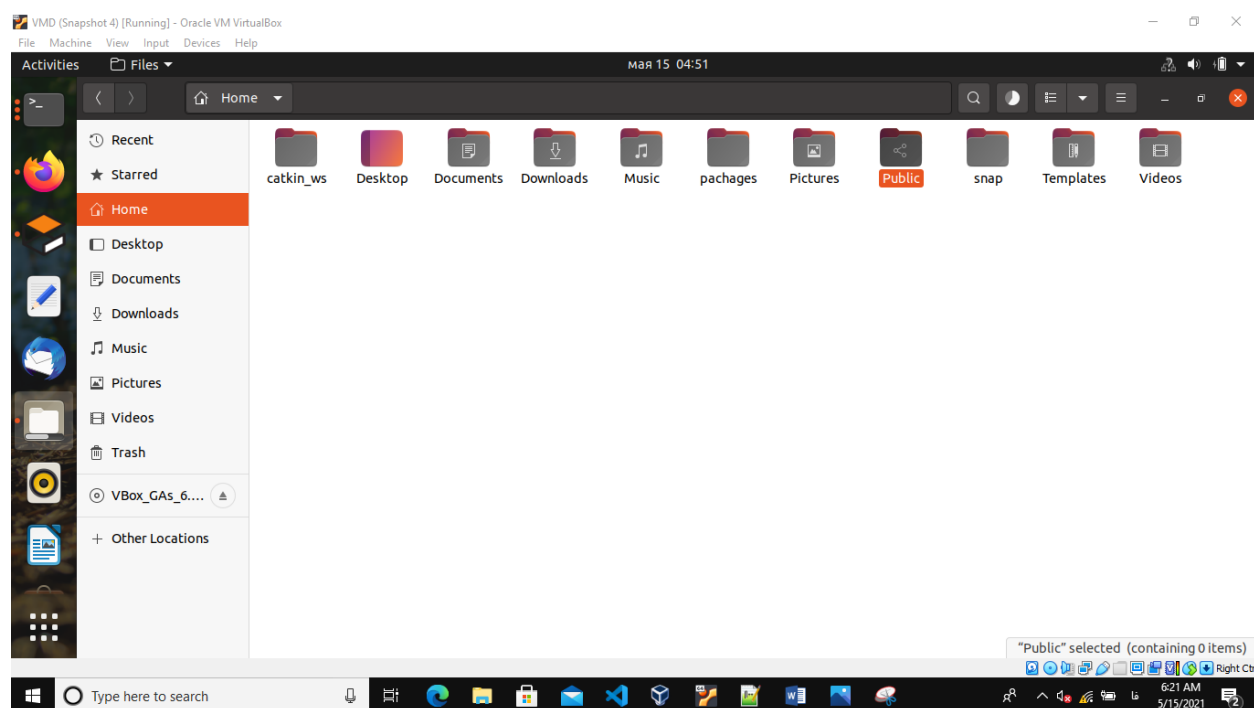
محمدحسین بدیعی  
شماره دانشجویی 9531701

استاد : دکتر مهدی جوانمردی  
بهار 1400

در ابتدا به توضیحات پیاده سازی مقاله می پردازیم و سپس به قسمت اصلی تمرین یعنی دنبال کردن یک مسیر مربعی توسط ربات خواهیم پرداخت و نتایج را نشان خواهیم داد.

## (مقاله)

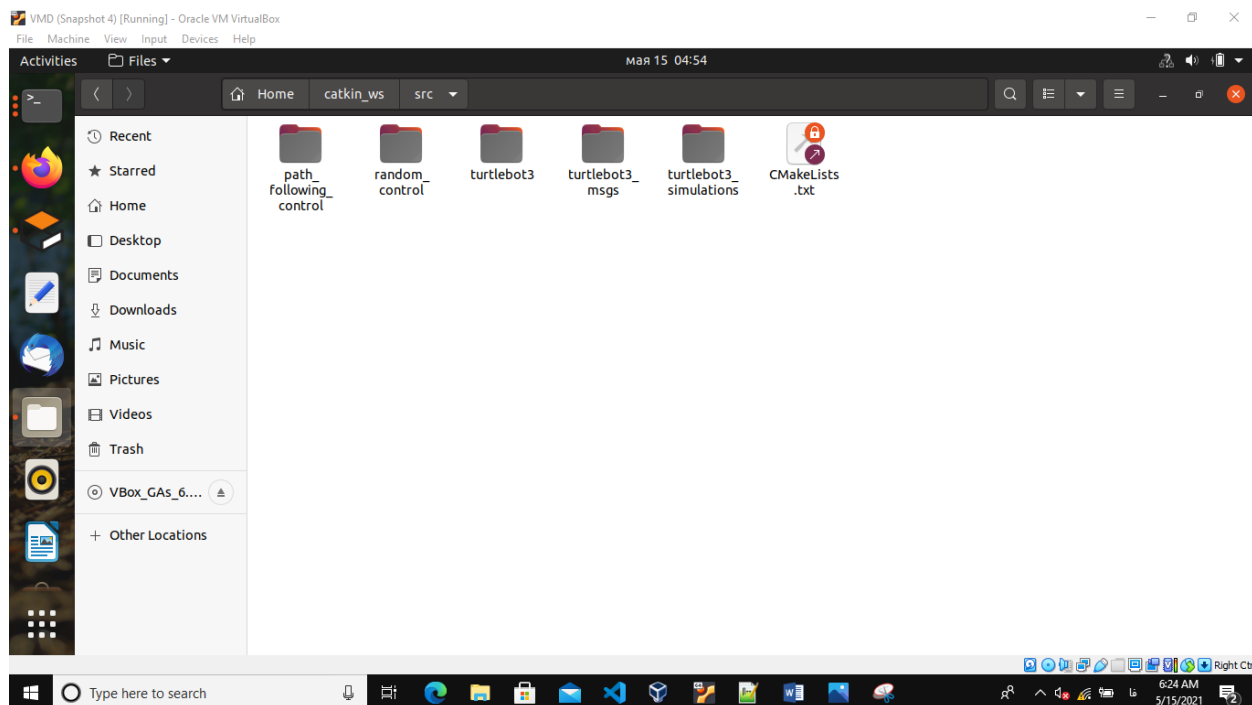
در ابتدای کار یک مقاله داده شده بود که در آن یک package با نام random\_control که متشکل از سه نود است را پیاده سازی کرده بود. بدین صورت که در ابتدا یک workspace با نام catkin را درست نموده و آن را make کرده و سپس در قسمت src مربوط به این workspace فایل های پایتون مربوط به هر نود را implement نموده است. ما نیز مطابق با مقاله همین کار را انجام دادیم.



در بالا می توانید فولدر مربوط به catkin را مشاهده بفرمایید.

همچنین در قسمت src نیز package های مربوطه را که در این workspace بکار می بریم را تعریف کرده و پیاده سازی یا در صورت آماده بودن مثل package مربوط به turtlebot3 از گیت دانلود می کنیم.

در زیر می توانید package ها را مشاهده کنید.



پکیج `path_following_control` مربوط به بخش پیاده‌سازی در قسمت بعد است که باید ربات یک مسیر مربعی را دانلود کند و خودمان با الگو گرفتن از `random_control` این پکیج و نودهای مربوطه اش را پیاده کرده ایم (این پکیج شامل دو نود `path` و `move_robot` می‌باشد) و همچنین پکیج `random_control` نیز مربوط به مقاله است که در ابتدای یادگیری آن را مطالعه و طبق کدهای موجود در مقاله پیاده‌سازی نمودیم. سه پکیج دیگر مربوط به `turtlebot3` هستند که با توجه به نسخه‌ی نرم‌افزار راس خود برای `turtlebot` می‌بایست از این پکیج استفاده می‌کردیم.

سپس طبق این مقاله سه فایل مربوط به پکیج `random_control` را نوشته و در این پکیج قرار داده و با لانچ کردن نرم‌افزار گزبو و ران گرفتن از این سه نود توانستیم یک ربات را که در صفحه با سرعت خطی و زاویه‌ای رندوم در دو بعد کف صفحه حرکت می‌کند را مشاهده کنیم. نوع مدلی که برای بخش مقاله استفاده کردیم `waffle` بود که می‌بایست کانفیگ‌های مربوطه را در فایل `bashrc` انجام می‌دادیم. البته در قسمت دوم از مدل `empty-world` یا `burger` استفاده نمودیم.

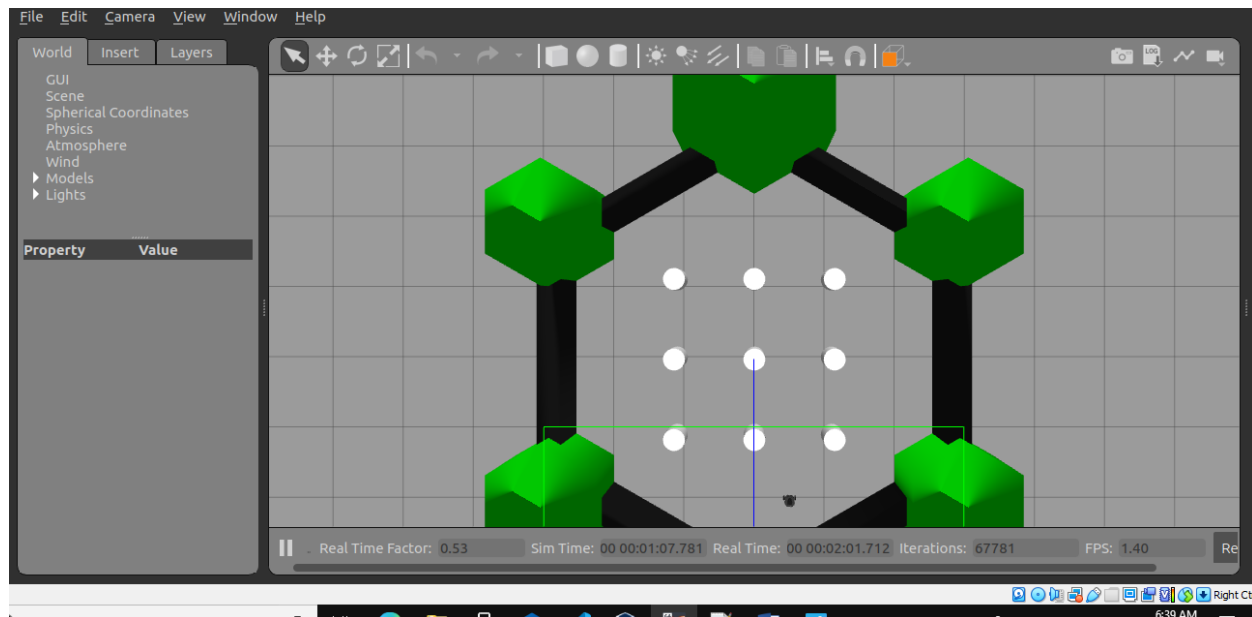
: Waffle

```
done
mohammad@mohammad-VirtualBox:~$ roslaunch turtlebot3_gazebo turtlebot3_world.lau
nch
```

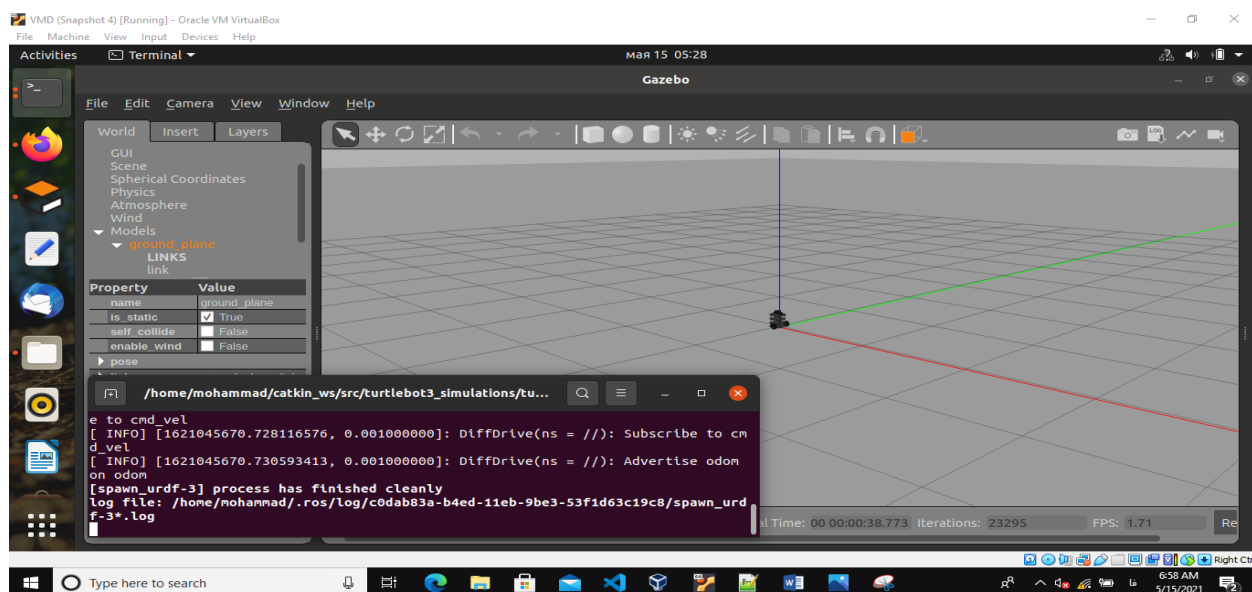
: Empty\_wold

```
/home/mohammad/catkin_ws/src/turtlebot3_simulations/tu...
bash: /home/akoubaa/catkin_ws/devel/setup.bash: No such file or directory
mohammad@mohammad-VirtualBox:~$ . ~/catkin_ws/devel/setup.bash
mohammad@mohammad-VirtualBox:~$ roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch
.. logging to /home/mohammad/.ros/log/c0dab83a-b4ed-11eb-9be3-53f1d63c19c8/rosl...
```

در مدل world نتیجه به صورت زیر است.



و empty\_world نیز که در بخش بعدی آشنا می‌شویم به صورت زیر است.



کدِ مربوط به `random_value` مقادیرِ تصادفی را برای سرعت خطی و زاویه‌ای ایجاد کرده (به ترتیب متر بر ثانیه و رادیان بر ثانیه) و سپس برای `move_robot` که وظیفه‌ی ارسال `command velocity` به رباتِ موجود در نرم‌افزارِ گزبو را دارد می‌فرستد. سپس از طریقِ `pose_monitor` نیز اطلاعاتِ `odometry` و موقعیتِ ربات نیز دریافت می‌شود که برای بخشِ دوم برای ما کاربرد دارد.

همچنین لازم به ذکر است که نحوه‌ی ارسال و دریافتِ پیام‌ها مشخص شود. در واقع این پیام‌ها به صورتِ `publish/subscribe` منتقل می‌شود. یعنی مثلاً گزبو اطلاعاتِ مربوط به ربات را `publish` می‌کند و نودِ `pose_monitor` در حالِ `subscribe` کردن است و به محضِ آنکه اطلاعات توسطِ سرور که در اینجا `roscore` است دریافت شد، آن را برای `subscriber` ارسال می‌کند. لذا سرورِ مربوط به راس همیشه باید در ابتدا بالا باشد که با دستورِ ساده‌ی `roscore` این قابلیت فراهم می‌شود.

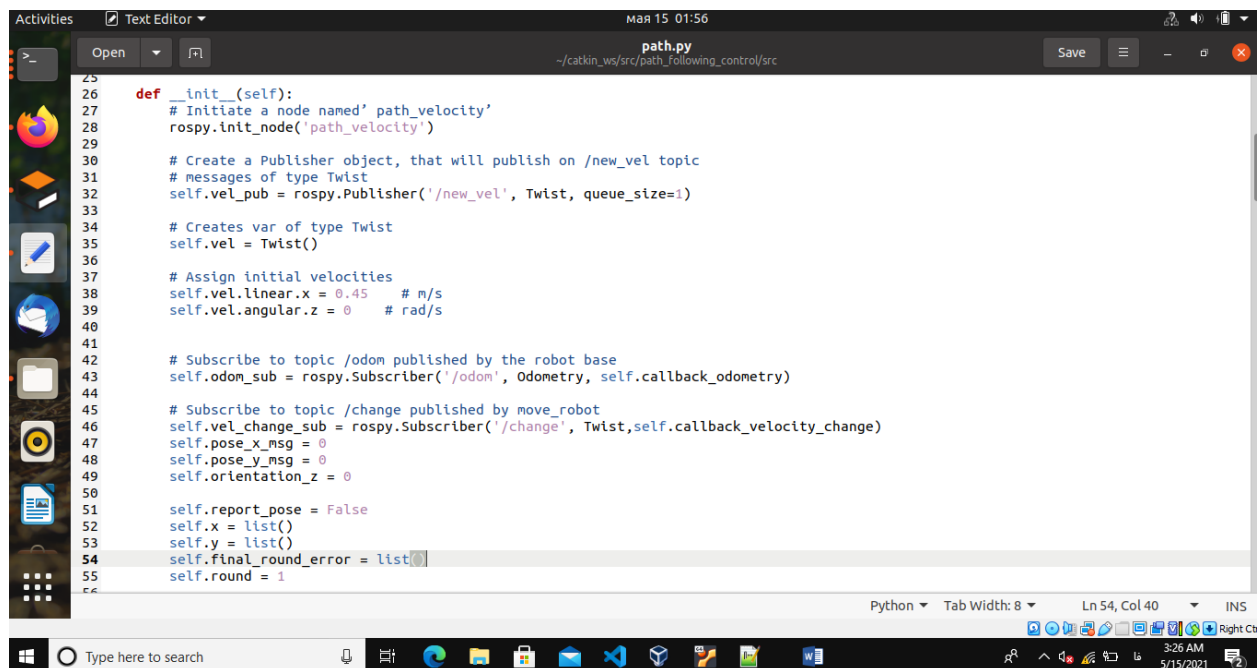
این روند برای ارسالِ داده‌های هدایتی برای کنترلِ سرعت خطی و زاویه‌ای ربات از طرفِ نودِ `random_values` هم صادق است و با همین متد تمامی این ارتباطات با سرعتِ بالا (پروتکل بسیار سبک است) شکل می‌گیرد. (توجه داشته باشید که نودهای سابسکرایبر به صورتِ تعریفِ یک تابعِ `callback` منتظرِ دریافتِ داده می‌مانند). در آخر مقاله نیز هم با دستوراتِ مختلفِ کنترلِ ربات از طریقِ کیبورد و هم سنسورهای `onboard` آشنا شدیم. حال به سراغِ توضیحِ قسمتِ بعدی که `path following` است، می‌روم.

## (دنبال کردنِ مسیرِ مربعی توسطِ ربات)

برای پیاده سازی این بخش دو نود را پیاده سازی کردیم. یکی از نودها وظیفه‌ی حرکت دادنِ ربات را بر عهده داشت که دقیقاً از همان فایلِ موجود در مقاله استفاده کردیم و نودِ دوم دو وظیفه اصلی را برعهده دارد، یکی اینکه اطلاعاتِ ربات را `subscribe` کرده و سپس متناسب با این اطلاعاتِ دریافتی اقدام به هدایتِ ربات در مسیرِ مناسب نماید.

اسم فایل دوم را `path.py` قرار دادیم و اسم نود را با نامِ `path_velocity` ، `initials` نمودیم.

ابتدا `init` کلاس را بررسی می‌کنیم.



```
25
26 def __init__(self):
27     # Initiate a node named 'path_velocity'
28     rospy.init_node('path_velocity')
29
30     # Create a Publisher object, that will publish on /new_vel topic
31     # messages of type Twist
32     self.vel_pub = rospy.Publisher('/new_vel', Twist, queue_size=1)
33
34     # Creates var of type Twist
35     self.vel = Twist()
36
37     # Assign initial velocities
38     self.vel.linear.x = 0.45 # m/s
39     self.vel.angular.z = 0 # rad/s
40
41
42     # Subscribe to topic /odom published by the robot base
43     self.odom_sub = rospy.Subscriber('/odom', Odometry, self.callback_odometry)
44
45     # Subscribe to topic /change published by move_robot
46     self.vel_change_sub = rospy.Subscriber('/change', Twist, self.callback_velocity_change)
47     self.pose_x_msg = 0
48     self.pose_y_msg = 0
49     self.orientation_z = 0
50
51     self.report_pose = False
52     self.x = list()
53     self.y = list()
54     self.final_round_error = list()
55     self.round = 1
56
```

همانطور که مشاهده می‌کنید، علاوه بر پارامترهای قبلی ای که در فایل موجود در مقاله داشتیم چند پارامتر دیگر را تعریف نمودیم. سه پارامتر مربوط به مختصات‌های مورد نیاز ما برای هدایت ربات است. این سه پارامتر، یکی موقعیت  $X$ ، دیگری موقعیت  $Y$  و آخرین پارامتر مورد نیاز ما زاویه فعلی ربات بوده که آنها را به ترتیب به صورت‌های زیر نامیدیم:

✓ موقعیت ربات در محور  $X$ : `pose_x_msg`

✓ موقعیت ربات در محور  $Y$ : `pose_y_msg`

✓ زاویه ربات نسبت به محور  $Z$ : `orientation_z`

سپس لیست‌های `self.x` و `self.y` را برای ذخیره‌ی مسیر ربات تعریف نمودیم. در نهایت برای ذخیره‌سازی خطای راند آخر ربات نسبت به مسیر اصلی، یک لیست دیگر با نام `final_round_error` تعریف کردیم. همچنین تعداد راندهایی که ربات مسیر را پیمایش می‌کند را نیز `round` نامیدیم.

پارامترهای و تابع مورد نیاز برای `subscribe` کردن مختصات را از فایل `pose_monitor` به فایل `path` انتقال دادیم.

تابع هدایت ربات را در `path_valuse` پیاده نمودیم. در این تابع صفحه مختصات به پنج بخش تقسیم شده‌اند. بخش اول مربوط به  $x \geq 1.1, y \leq 1.1$  است. دلیل این امر که ما بجای استفاده از نقاط 1.5 که نقاط گوشه‌ی مربع هستند، از 1.1 استفاده کردیم این بوده که ربات هنگام چرخش مسافتی را می‌پیماید و طبیعتاً برای جبران این مسافت پیموده شده قادر کمتری را استفاده کردیم.

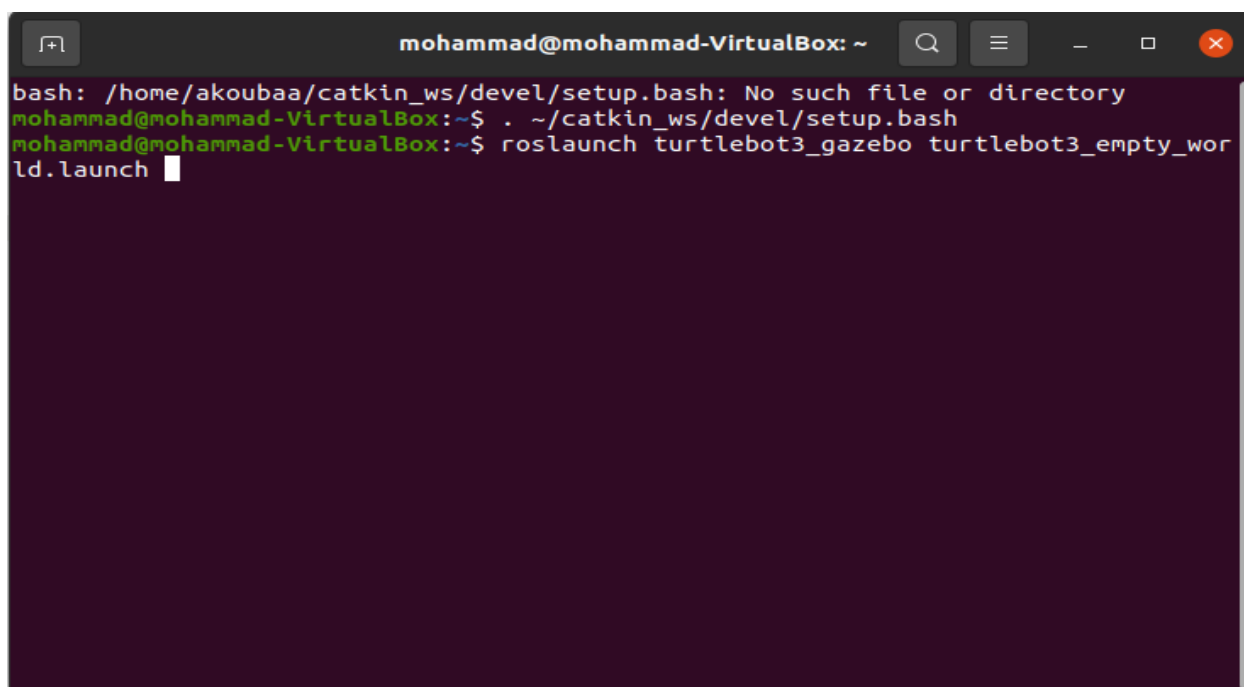
بخش دوم مربوط به  $x \geq -1.1, y \geq 1.1$  است. و همینطور بقیه‌ی اطراف مربع، بخش‌های سوم و چهارم هستند که توابع هدایت ربات را برای هر کدام با تنظیم نسبتا مناسب زاویه چرخش و سرعت، پیاده کرده ایم. بخش پنجمی که ذکر نکردیم مربوط به داخل مربع است. در این حالت به ربات اجازه دادیم تا با یک سرعت اولیه‌ای از مربع خارج شده و خود را به یکی از نواحی فوق رساند. وقتی وارد یکی از چهار ناحیه‌ی فوق شد، آنگاه متناسب با آن ناحیه، دستور هدایت ربات از path به نودِ move ارسال می‌شود.

دقت کنید که چون ربات با سرعت مورد نظر در سوال در محیط گزبو خارج می‌شود و عملا پیمایش یک مربع کوچک با این سرعت، مسیر ربات را مختل می‌کند. لذا ما سرعت را نصف سرعت مورد نظر در سوال یعنی 0.45 متر بر ثانیه برای سرعت خطی در نظر گرفتیم. توجه داشته باشید که تنها در صورتی که ربات مسیر درست را پیمایش کند حق دارد از این سرعت طبیعت کند و در صورتی که به مسیر دیگری منحرف شده باشد، سرعتش را کم نمودیم تا و زاویه را تغییر دادیم مادامی که به مسیر اصلی باز گردد.

ابتدا عملکرد ربات در محیط گزبو را می‌بینیم و بعد به سراغ نتایج شبیه سازی می‌رویم.

نقطه شروع ربات را در مرکز مربع یعنی (0,0) گرفتیم. لذا ربات می‌بایست خود را به مربع رسانده و آن را پیمایش کند.

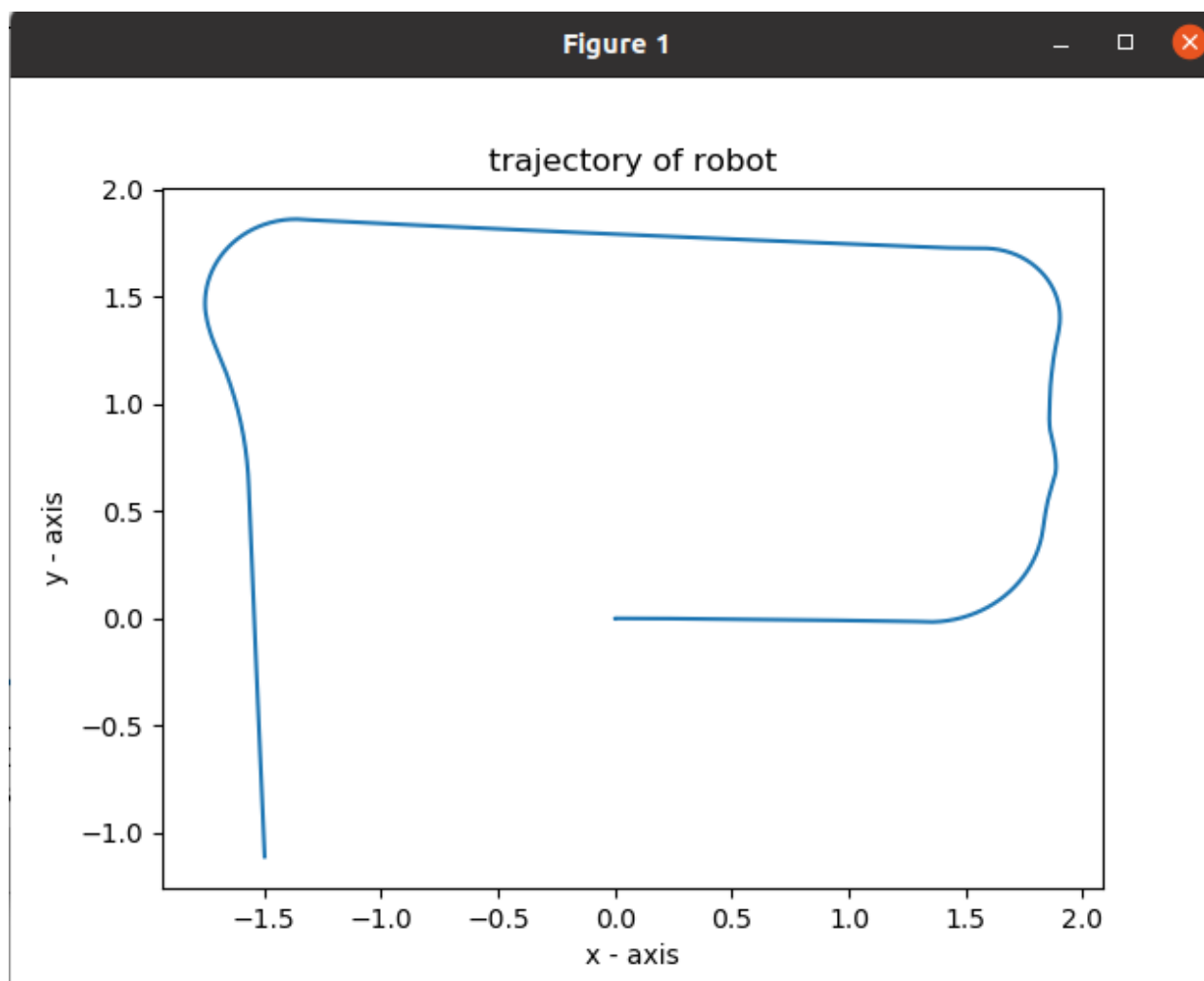
طبق خواسته ی سوال empty-world را بر روی گزبو لانچ می‌کنیم.



```
mohammad@mohammad-VirtualBox: ~  
bash: /home/akoubaa/catkin_ws/devel/setup.bash: No such file or directory  
mohammad@mohammad-VirtualBox:~$ . ~/catkin_ws/devel/setup.bash  
mohammad@mohammad-VirtualBox:~$ roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch
```

دقت کنید که نقطه پایانی هر راند برای ربات را نقطه  $(-1.5, -1.5)$  در نظر گرفته و در صورتی که ربات خود را به این نقطه برساند، عملاً یک راند را پیمایش کرده است.

ابتدا trajectory ربات را برای یک راند، بررسی می‌کنیم.

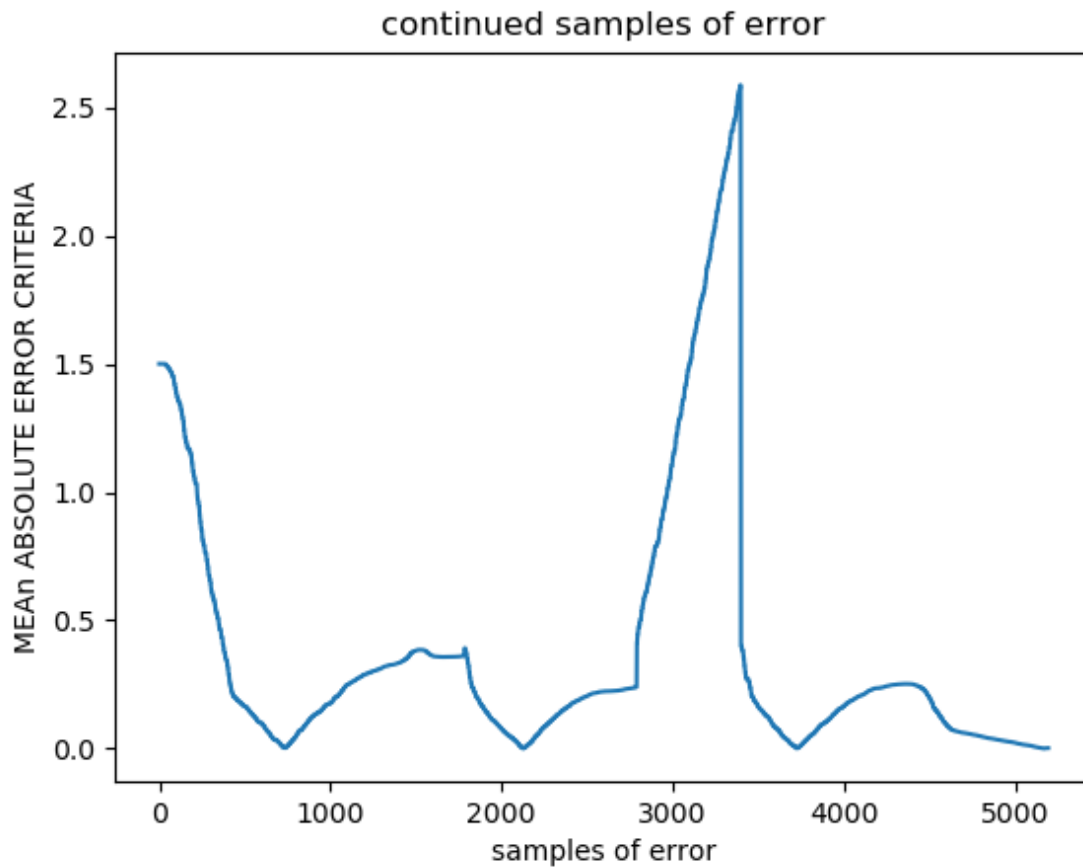


همانطوری که مشاهده می‌کنید، این ربات از مرکز مربع یعنی  $(0,0)$  شروع به حرکت کرده است و پس از رساندن خود با مربع، یک راند را که رسیدن تا نقطه‌ای  $(-1.5, -1.5)$  بوده است را پیموده.

خطای این ربات در راند فوق به صورت زیر است. معیار خطای که در نظر گرفتیم، خطای MAE یا همان خطای مطلق بوده است که به صورت زیر می‌باشد.



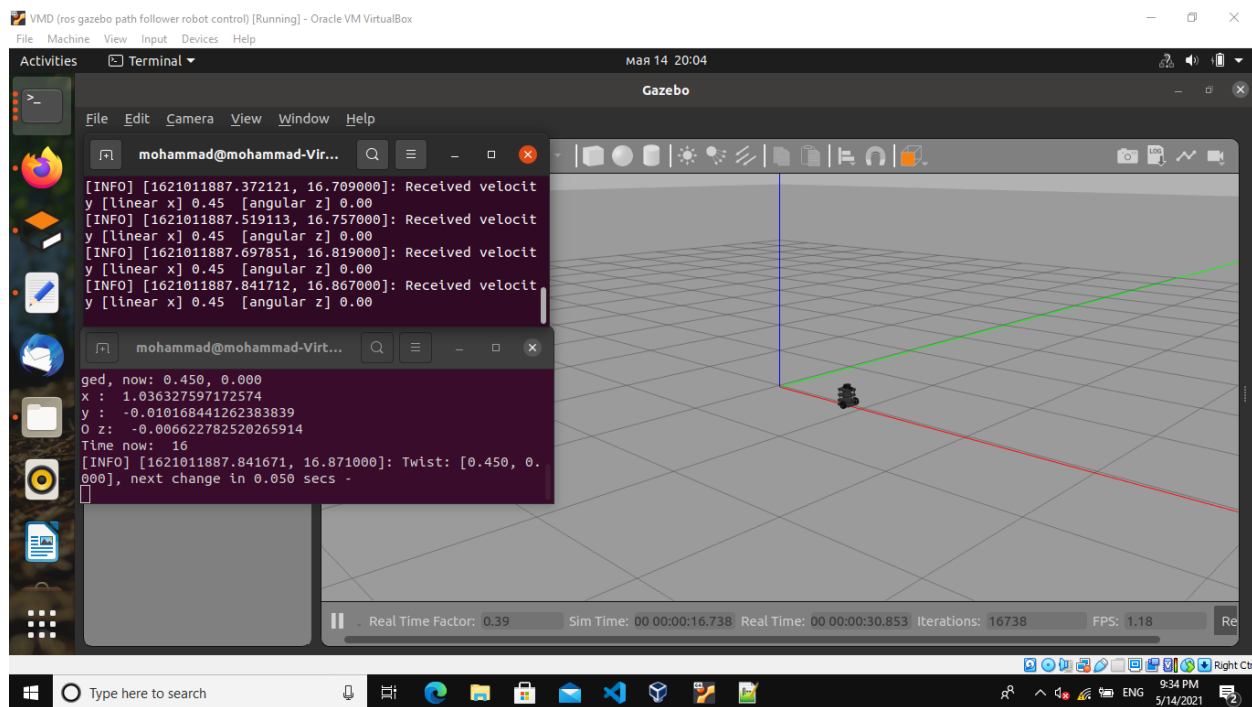
Figure 2



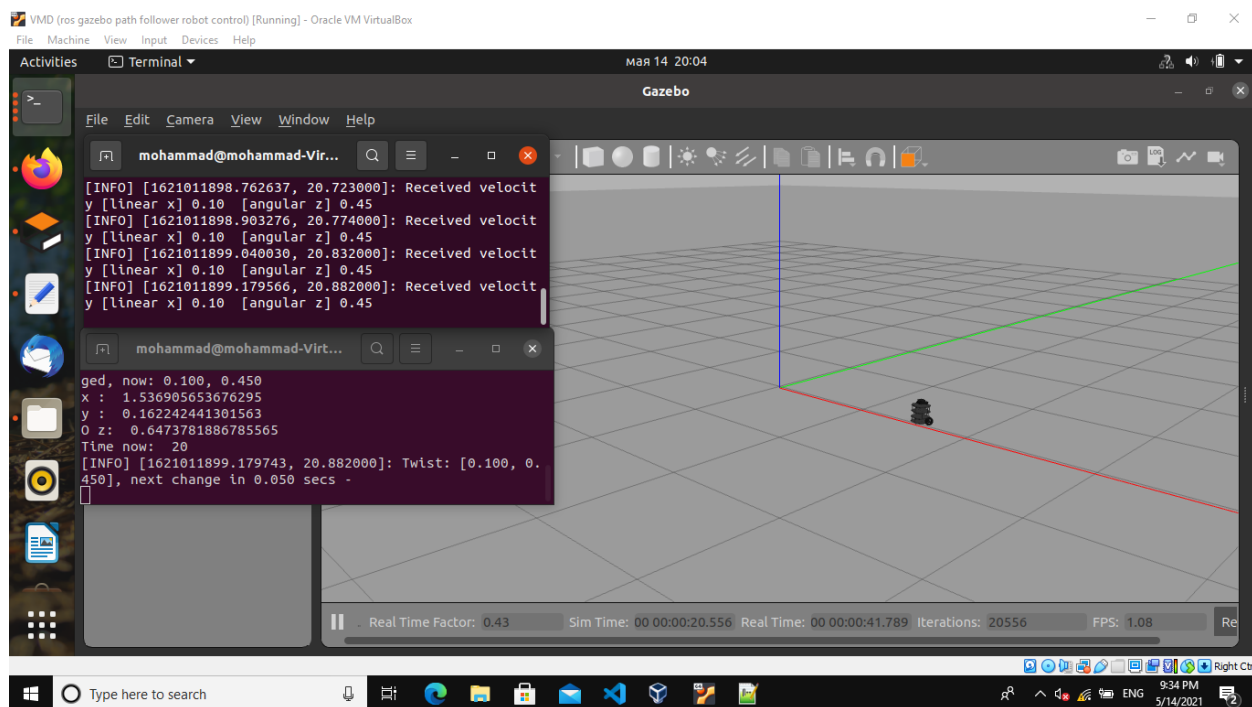
همانطور که مشاهده می‌کنید، این ربات برای یک راند نسبتاً خوب عمل می‌کند و می‌تواند خود را روی مسیر به مقصد نهایی برساند. به نظر من این خروجی از ربات قابل قبول است جز در نقاط گوشه که ربات با افزایش ناگهانی خطا مواجه می‌شود. من برای حل این مشکل همانطور که عرض کردم، ربات را در موقعیت‌های مکانی نزدیکتر، قبل از رسیدن به نقطه‌ی گوشه چرخش دادم و به ضلع دیگر هدایت کردم که نتیجه نسبتاً مطلوب است.

قبل از بررسی برای 10 راند و البته بدست آوردن خطای آخرین راند، می‌خواهیم موقعیت ربات را در گزبو به شما نشان دهیم.

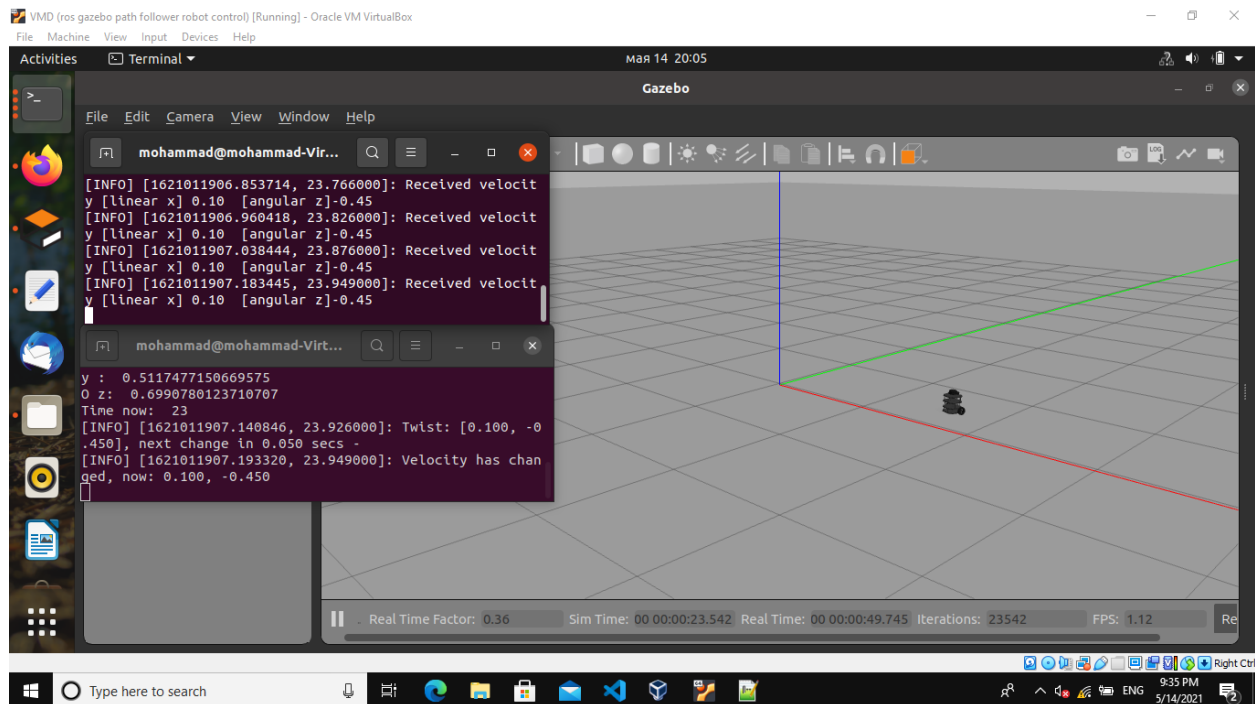
ربات از  $(0,0)$  به سمت  $(1.5,0)$  در حرکت :



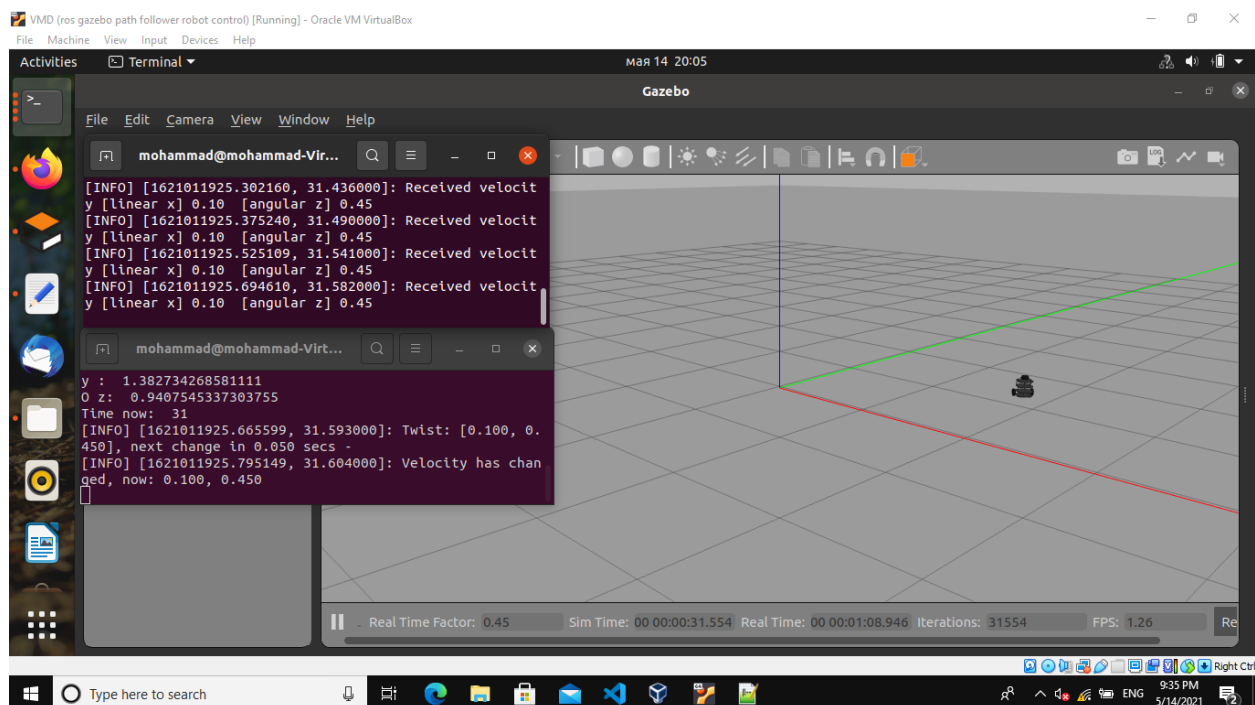
ربات در نقطه  $(0, 1.5)$  در حال تغییر جهت :



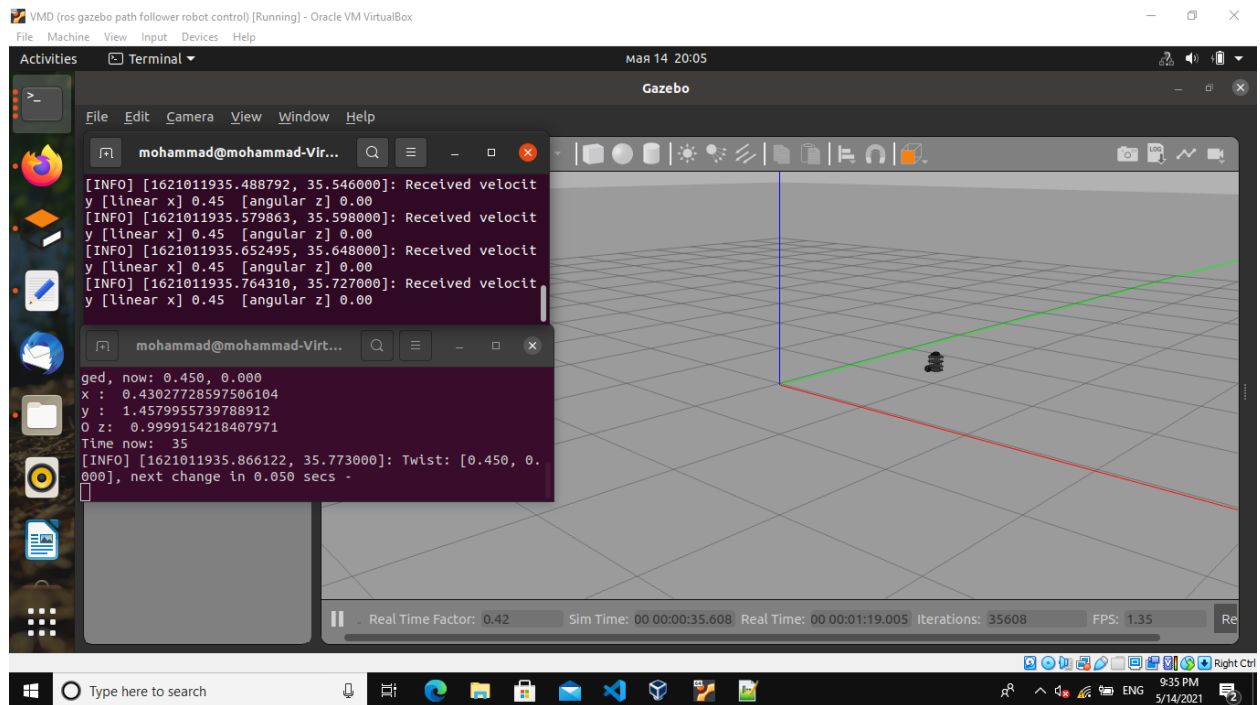
ربات بر روی  $x=1.5$  در حال حرکت به سمت نقطه‌ی  $(1.5, 1.5)$ :



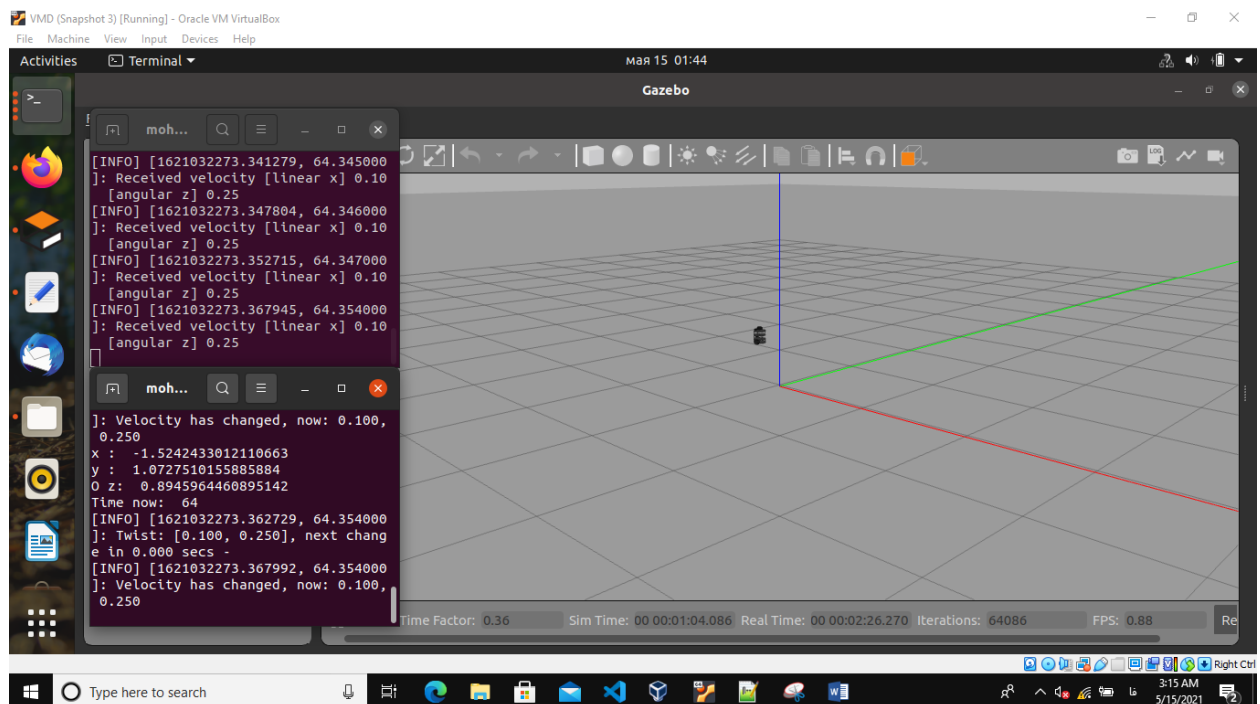
ربات در نقطه‌ی  $(1.5, 1.5)$  در حال تغییر جهت:



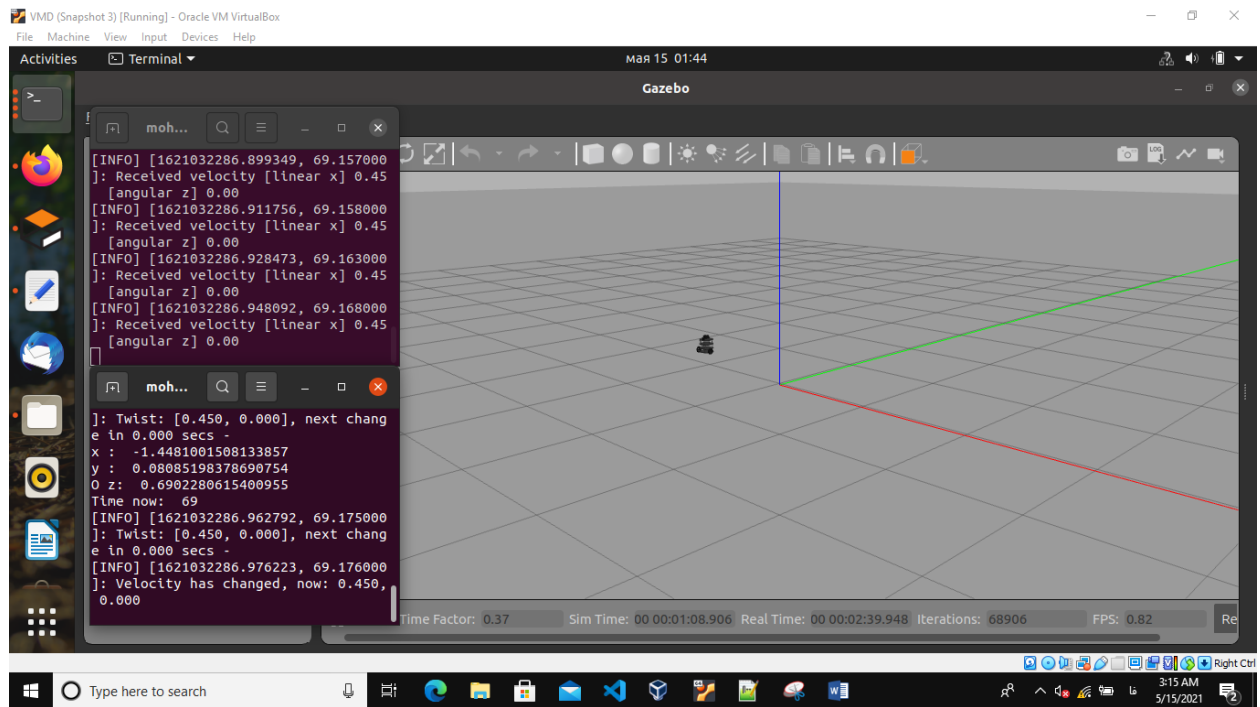
ربات بر روی  $y=1.5$  به سمت نقطه‌ی  $(-1.5, 1.5)$  در حال حرکت:



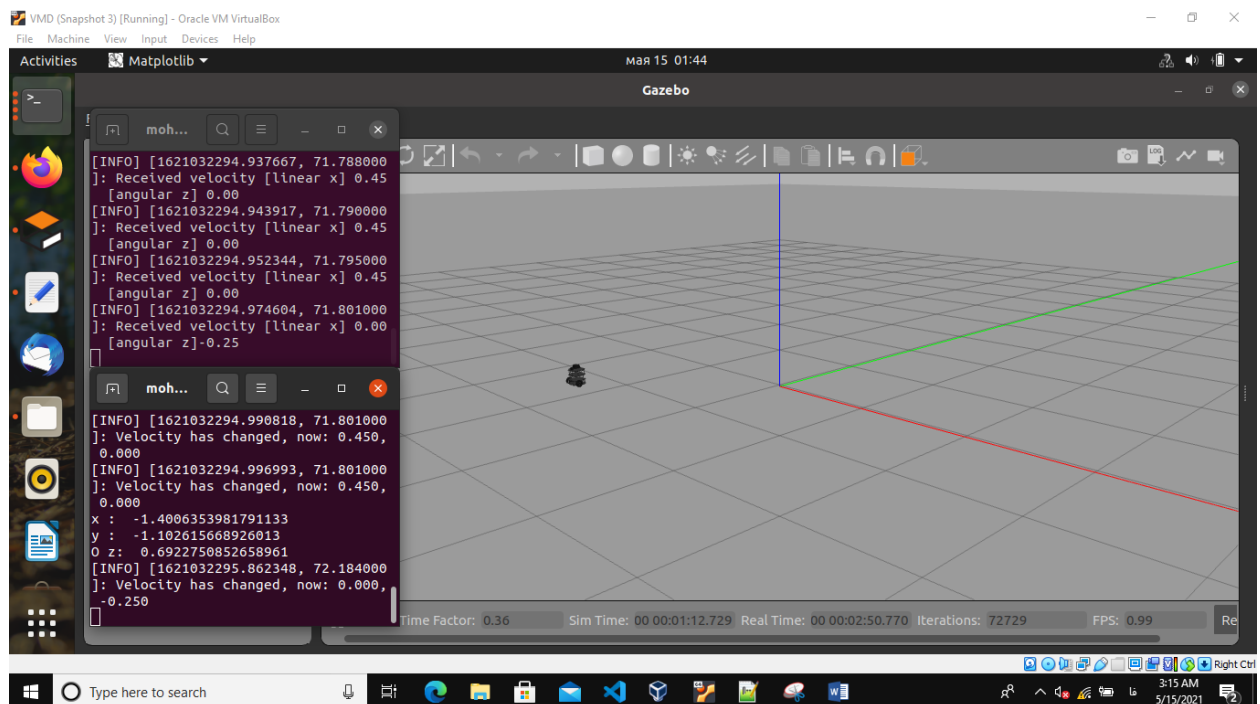
ربات در نقطه (1.5, -1.5) در حال تغییر جهت :



ربات بر روی  $x=-1.5$  در حال حرکت به سمت نقطه (-1.5, -1.5) :



رسیدن به مقصد راند اول یعنی نقطه  $(-1.5, -1.5)$  که گوشه‌ی سمت چپ پایین مربع می‌باشد:

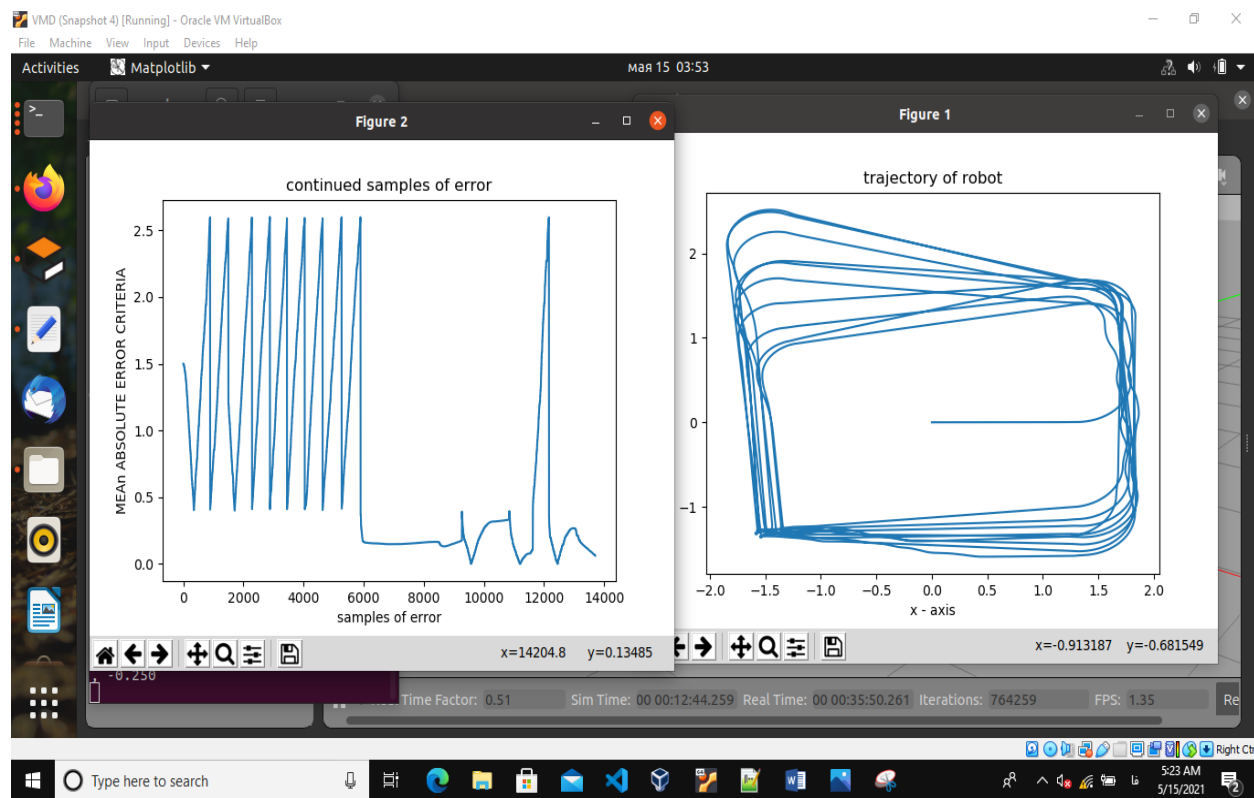


حال اینکار را طبق خواسته‌ی سوال برای 10 راند انجام می‌دهیم و trajectory آن را رسم کرده و خطای راند آخر آن را نمایش می‌دهیم.

پارامتر راند را از یک به ده تغییر می‌دهیم:

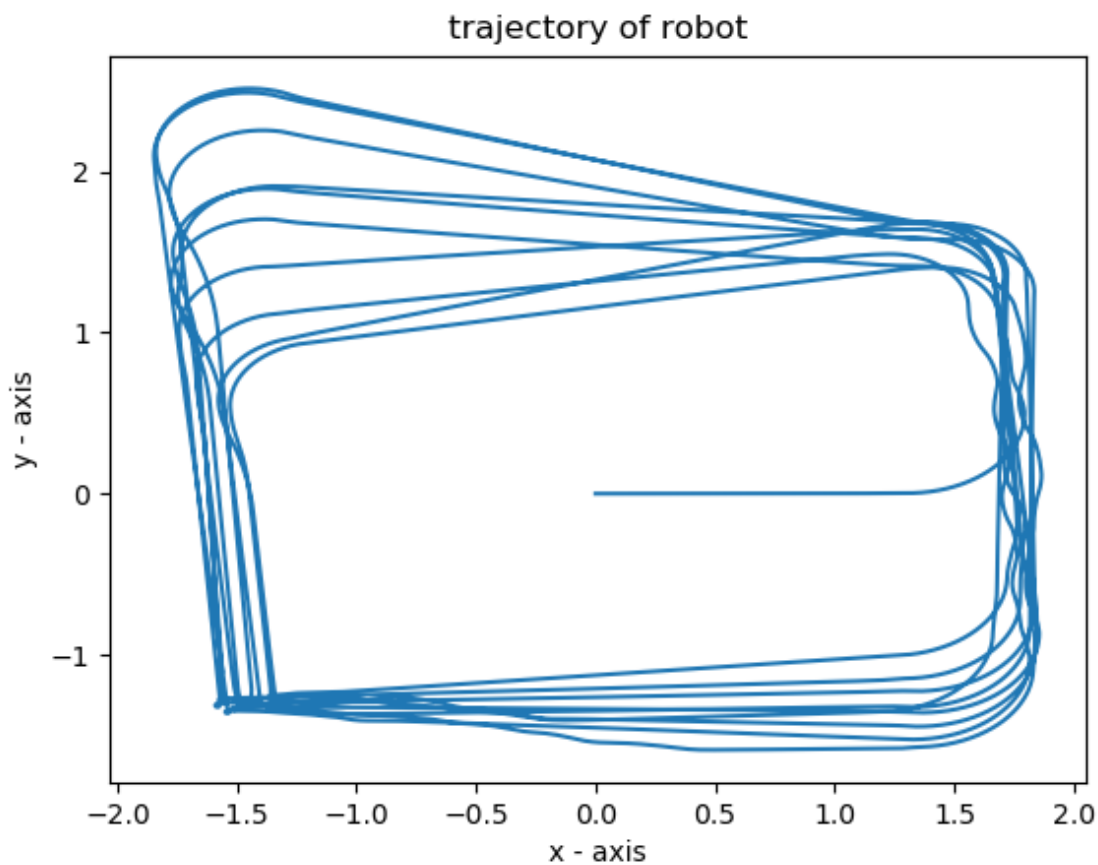
```
51 self.report_pose = False
52 self.x = list()
53 self.y = list()
54 self.final_round_error = list()
55 self.round = 10
56
57
58
```

پس از حدود 12 دقیقه شبیه سازی، ربات 10 راند را پیمود.



ابتدا trajectory را بررسی می‌کنیم.

Figure 1

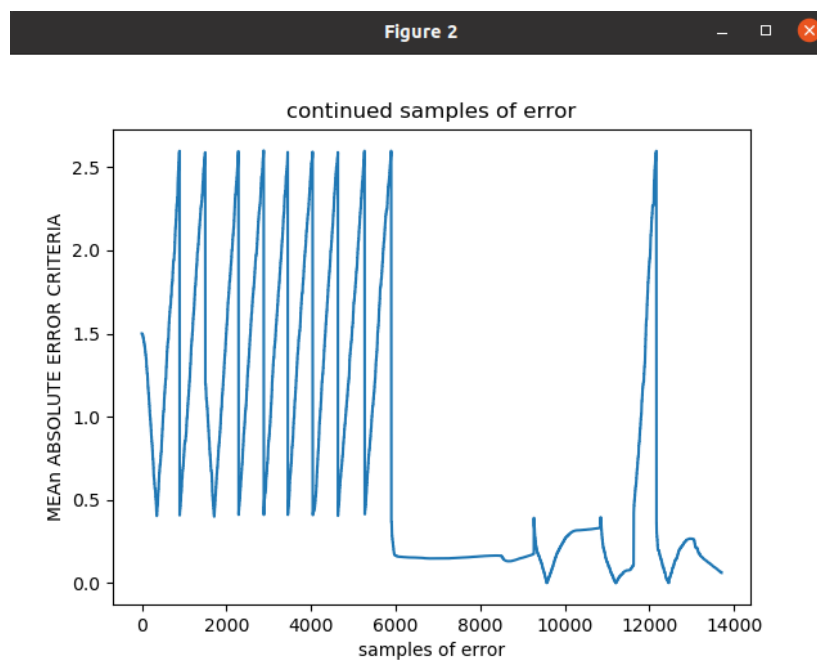


همانطور که مشاهده می‌کنید، ربات از نقطه  $(0,0)$  یعنی واقع در مرکز مربع حرکت خود را آغاز کرده است. سپس خود را به مربع با ضلع سه سانت و مرکز  $(0,0)$  رسانده و تا مقصد یعنی نقطه  $(-1.5, -1.5)$  که در ابتدا این نقطه را مقصد فرض کردیم، به تعداد 10 راند به آن رسیده و مجدداً از آن عبور کرده و باز مسیر مربعی را طی نموده و مجدداً خود را برای 10 بار به این نقطه رسانده. پس از بار دهم نیز در این نقطه متوقف شده است.

در رابطه با این trajectory باید بگوییم که ما در سه ضلع مربع، در هر ده مرحله، پیمایش بسیار مناسبی را برای ربات داشتیم. اما ربات در ضلع بالایی بر روی مسیر به خوبی هدایت نشده است. دلیل این رخداد آن است که پارامتر دریافتی تنها بر روی این ضلع نزدیک به یک بوده و چه حرکت ساعتگرد و چه حرکت پادساعتگرد روی آن، باعث کاهش مقدار این پارامتر می‌شده و ما درک خوبی نسبت به زاویه بر روی این ضلع نداشتیم و مسلماً هم نتوانستیم کنترل خوبی را روی ضلع بالایی مسیر داشته باشیم. در واقع عدم تشخیص کاهش یا افزایشی بودن زاویه ربات بر روی این ضلع باعث شد که نتوانیم هدایت ربات را به خوبی انجام دهیم. با این

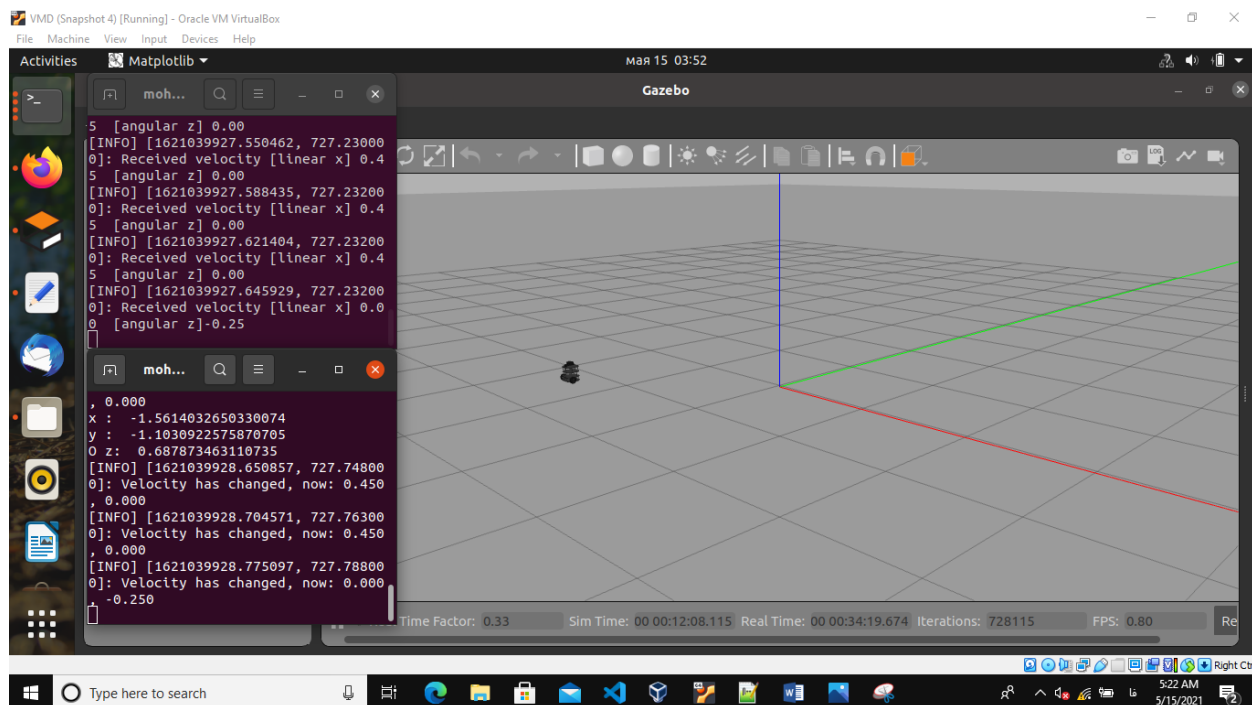
حال من سعی کردم که زاویه و سرعت ورود به این ضلع را برای ربات به گونه‌ای تنظیم نمایم که در یک رفتار مناسب از خود نشان دهد و اگر به نمودار توجه بفرمایید مشاهده می‌کنید که با تقریب خوبی، میانگین تمامی این ده مسیر، بر روی  $y=+1.5$  قرار دارد.

نمودار بعدی مربوط به خطای راند آخر در پیمایش ربات است. نکته قابل توجه این است که خطای موجود در trajectory فوق برای ربات تقریباً نزدیک به صفر است ولی ما در نمودار زیر تنها در بازه‌ی دوم این مطلب را مشاهده می‌کنیم. در واقع دلیل این امر آن است که اتمام ران مصادف با شروع شدن راند بعدی نیست. در واقع وقتی راند قبلی برای ربات به اتمام می‌رسد، ربات زاویه‌ی خود را برای راند جدید آماده کرده و وقتی زاویه ربات در حالت تتا برابر با صفر قرار گرفت، ربات حرکت خود را در راند جدید آغاز می‌کند. لذا این مساله فقط به تعیف بازه‌ی شروع و اتمام هر راند بوده و فقط نیمه‌ی دوم نمودار زیر مربوط به راند آخر است. توجه کنید که این مساله بدلیل این مشکل بوجود آمد که ما می‌بایست به گونه‌ای هر بار مسیر پیمایش شده را تشخیص می‌دادیم و یک حالت توقف برای دهمین بار تعریف می‌کردیم. لذا تفکیک راند جدید و قبلی، بدون در نظر گرفتن رخداد خاصی در رفتار ربات ممکن نبود. لذا من رسیدن به نقطه  $(-1.5, -1.5)$  را نقطه‌ی پایان راند و برگشت ربات به زاویه‌ی مناسب را شروع راند گرفتم زیرا ربات مجبور از  $\pi/2$  تغییر زاویه دهد و با این تفاوت فاحش به راحتی می‌توانستم راند قبلی را از راند جدید تفکیک نمایم. لذا به عبارتی داده‌های نوک تیز بدلیل undefined بودن مکان ربات در راند یا حتی عدم انجام عملیات subscribe هستند و در واقع جزء outlier های توزیع خطا به حساب می‌آیند.



در نهایت نیز ربات پس از 12 دقیقه شبیه‌سازی به نقطه  $(-1.5, -1.5)$  برای بار 10ام رسید و ده بار پیمایش خود را تکمیل کرد.





هر مربع کوچک دارای ضلع یک متری است. پس ربات در حال حاضر پس از ده بار پیمایش مسیر در نقطه‌ی درت و تقریباً برابر با  $(-1.5, -1.5)$  قرار گرفته است.

دقت بفرمایید که دو پکیج که یکی مربوط به مقاله و دیگری مربوط به پیاده سازی مسیر مربعی که انجام داده ام را قرار دادم. همچنین با توجه کامندهای مختلفی که برای نسخه‌های مختلف راس تعبیه شده است. از پکیج های turtlebot3 مربوط به ROS noetic برای اجرای turtlebot در هنگام لانچ گزبو استفاده کنید. (آخرین لینک مربوط به نصب 3 پکیج turtlebot3) منابع استفاده شده:

- [https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwihvMG518rwAhVdQUEAHct6Bp4QwqsBMAB6BAGDEAM&url=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3D\\_yBrwWXRiz0&usg=AOvVaw0XgujjgVbqE6iJjBBzZZTC](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwihvMG518rwAhVdQUEAHct6Bp4QwqsBMAB6BAGDEAM&url=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3D_yBrwWXRiz0&usg=AOvVaw0XgujjgVbqE6iJjBBzZZTC)
- <https://automaticaddison.com/how-to-launch-the-turtlebot3-simulation-with-ros/>
- <https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/#simulate-in-various-world>
- [https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=video&cd=&cad=rja&uact=8&ved=2ahUKWwjf4sDy18rwAhUZiFwKHZCoAdsQtwIwAXoECACQAw&url=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3Df\\_lcbVQ3Oa4&usg=AOvVaw1Ja2Z85L6h\\_VQLS6VNzg0M](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=video&cd=&cad=rja&uact=8&ved=2ahUKWwjf4sDy18rwAhUZiFwKHZCoAdsQtwIwAXoECACQAw&url=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3Df_lcbVQ3Oa4&usg=AOvVaw1Ja2Z85L6h_VQLS6VNzg0M)