|  |  |  |
| --- | --- | --- |
| C:\Users\mohammad\Downloads\image(1).png | به نام خدا | C:\Users\mohammad\Pictures\amirkabirLogo.png |
| **دانشگاه صنعتی امیرکبیر**  **دانشکده‌ مهندسی کامپیوتر**  **اصول علم ربات**  **تمرین سری اول** | | |

|  |  |
| --- | --- |
| مهدی رحمانی | نام و نام خانوادگی |
| 9731701 | شماره‌ دانشجویی |
|  | تاریخ ارسال گزارش |

­

**فهرست گزارش سوالات**

[بخش 1 – نصب نرم افزار 3](#_Toc94975479)

[سوال ۲ – عنوان سوال 4](#_Toc94975480)

[سوال 3 – عنوان سوال 5](#_Toc94975481)

# بخش 1 – نصب نرم افزار

ابتدا باتوجه به ویودیوی قرار داده شده، سعی میکنیم تا ROS را نصب کنیم. باتوجه به اینکه نسخه ROS مورد استفاده ROS-noetic میباشد، سیستم عامل پیشنهادی برای آن Ubuntu 20.04.5 LTS میباشد. پس در گام اول چنانچه این سیستم عامل را نداریم لازم است تا آن را نصب کنیم. برای این امر من از VM کمک گرفتم و یک ماشین مجازی ایجاد کرده و بر روی آن نسخه سیستم عامل پیشنهادی را نصب کردم.

همچنین سخت افزاری که به این VM اختصاص دادم در زیر آمده است:

* Allocated hard disk space : 40 GB
* RAM : 8 GB
* Number of processors : 2
* Number of cores per processor : 3

سپس در گام بعد باتوجه به مراحل آمده در لینک پیش رفته و ROS را نصب کردم. کدهای اجرا شده در ترمینال جهت نصب عبارتند از:

* Setup your sources.list
* sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb\_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
* Set up your keys
* sudo apt install curl # if you haven't already installed curl
* curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
* Installation
* sudo apt update
* sudo apt install ros-noetic-desktop-full
* Environment setup
* source /opt/ros/noetic/setup.bash
* echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
* source ~/.bashrc
* Dependencies for building packages
* sudo apt install python3-rosdep python3-rosinstall python3-rosinstall-generator python3-wstool build-essential
* Initialize rosdep
* sudo rosdep init
* rosdep update

# بخش ۲ – انجام تمرین هندزان از روی ویدئو

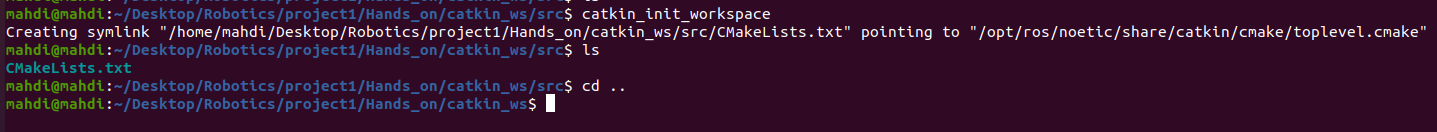
باتوجه به ویدیوی هندزان بارگذاری شده برای این ترم جهت آشنایی با مفاهیم اولیه و اساسی در ROS سعی شده تا مراحل را یک بار پا به پای ویدئو انجام دهیم.

Graphical user interface, text, application

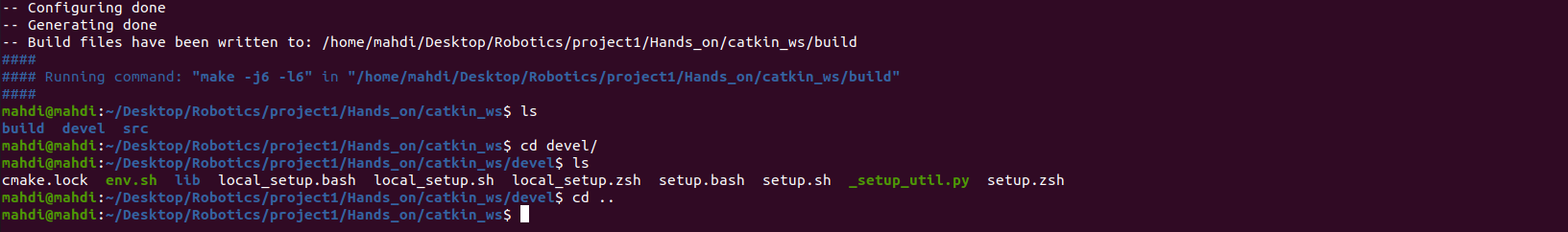
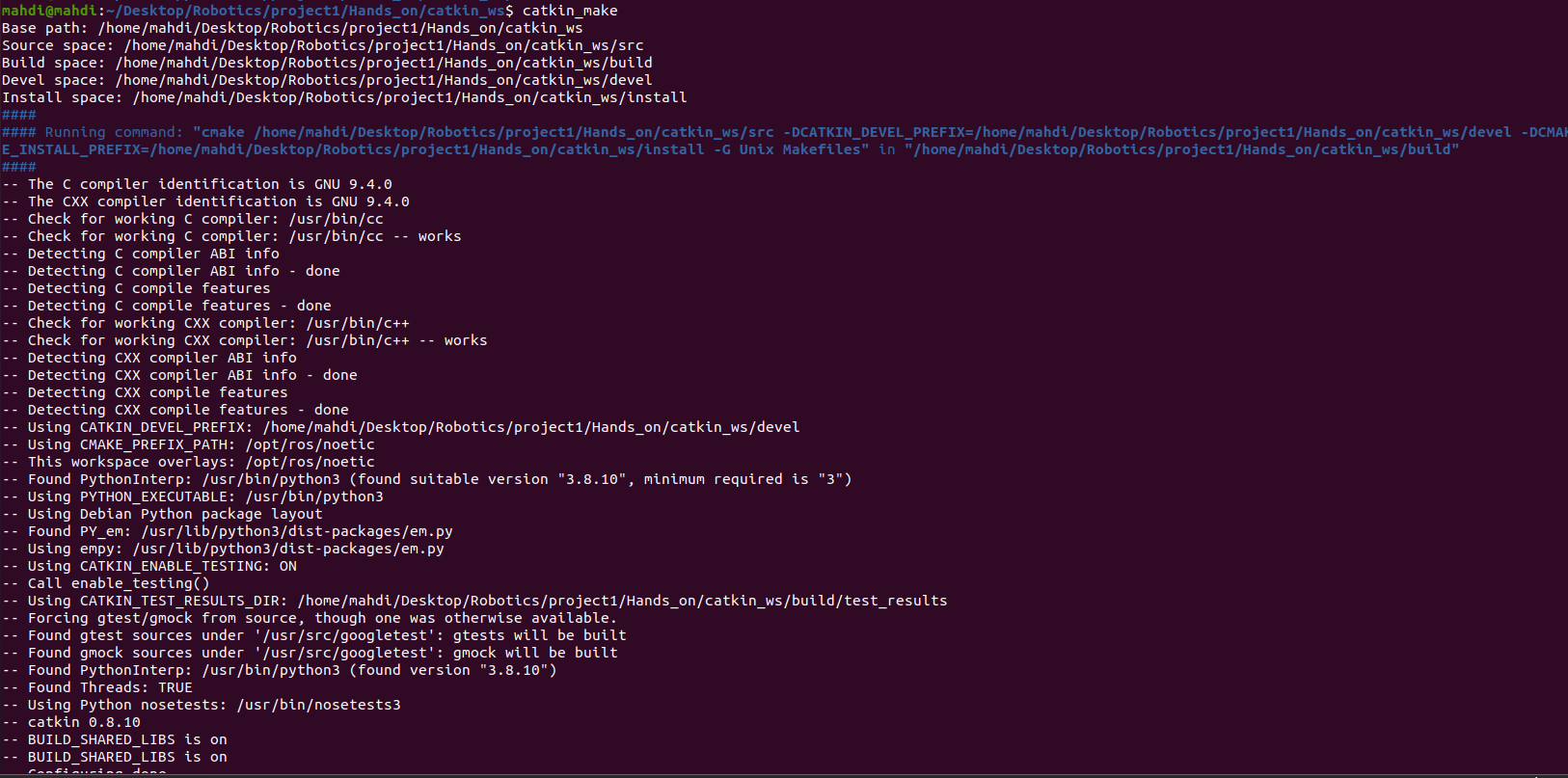
Description automatically generatedابتدا لازم است یک workspace درست کنیم. ابتدا لازم است به مسیری که میخواهیم package را در آن بسازیم برویم.

Text

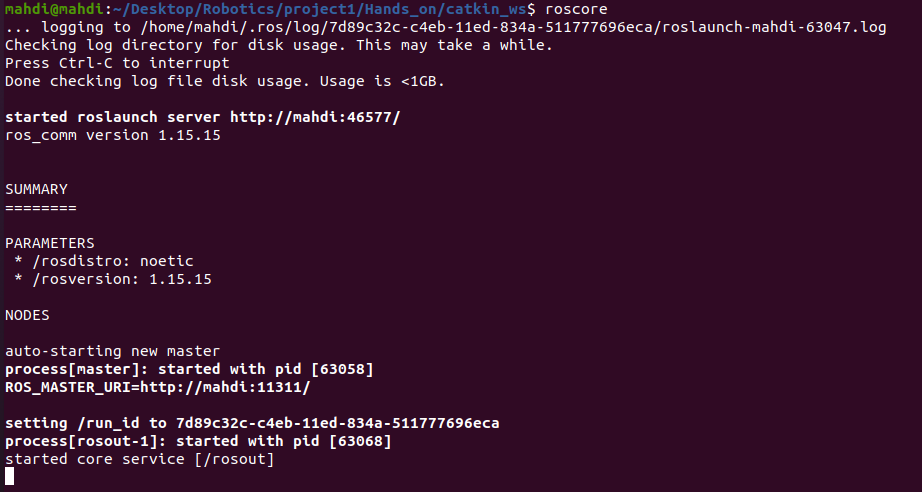
Description automatically generatedسپس به کمک دستور mkdir -p name/src پکیج را شروع به ساختن میکنیم. در جای name اسم دلخواه خود را میگذاریم. برای مثال catkin\_ws میباشد. پس از اجرا ساخته میشود و اگر وارد این فولدر شویم میبینیم یک فولدری به اسم src ساخته شده که ولی داخل آن فعلا چیزی وجود ندارد.

لازم است تا در این فولدر work space را initialize بکنیم. دستور مربوطه catkin\_init\_workspace میباشد. سپس ls میزنیم و میبینیم که CMakeLists.txt ایجاد شده است. در نهایت cd .. میزنیم که به پوشه قبلی برگردیم.

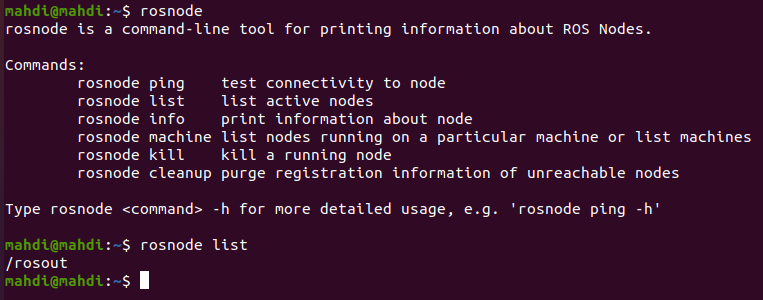
حال باید دستور catkin\_make را اجرا کنیم. و بعد اگر ls را بزنیم خواهیم دید دو فولدر build و devel اضافه شدند. ما همه‌ی کدها و نودها و چیزهایی که میسازیم را در پوشه src قرار میدهیم. این catkin\_make یک building tool میباشد و چیزهایی که توی src هست مثل نودها را در صورت لزوم در فولدرهای build و devel قرار میدهد. اگر مثلا داخل devel برویم میبینیم یک سری فایل و پوشه مثل lib در آن هست که بخش هایی از کدتان در آن قرار میگیرد. یک setup.bash هم اینجا داریم که اگر بخوایم با این workspace کار کنیم لازم است آن را source کنیم.

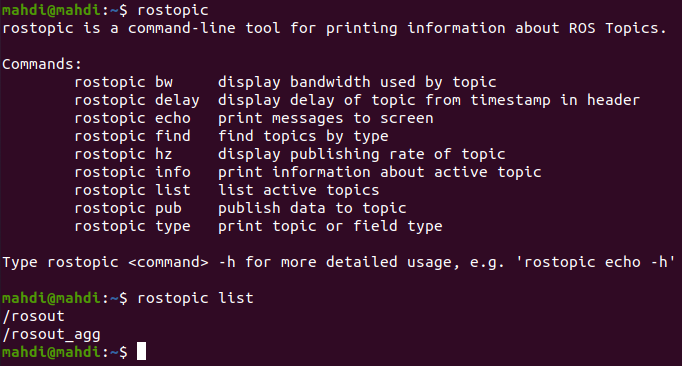


قبل از هرچیز لازم است وقتی که میخواهید نودی را ران بکنید، ROS Core را بالا بیاورید. ( همون ROS Master میباشد). وقتی ROS را نصب میکنید یک سری Packageها و نودها وجود دارند که واسه Tutorialهاش هستند و میتوانید از آن‌ها استفاده کنید. ماهم در ادامه از اینا استفاده میکنیم و ران میکنیم تا با نود و تاپیک اشنا شویم. برای این منظور در همین ترمینال roscore را اجرا کنیم:

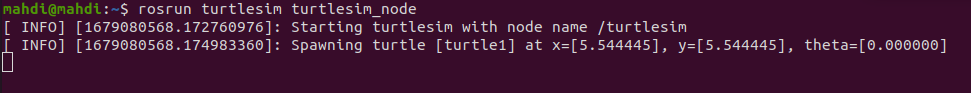


حال میتوان نودهای لازم را اجرا کرد. برای این منظور یک ترمینال جدید باز میکنیم. قبل از آن با یک سری دستورها آشنا میشویم. اگر در ترمینال جدید rosnode را بزنیم، کامندهای مربوط به node و توضیحات مروبط به آن‌ها را میتوان دید.

حال اگر rosnode list را بزنیم توقع داریم نود خاصی وجود نداشته باشد. ( یک نود که به صورت دیفالت هست به نام rosout وجود دارد.)

یک دستور دیگر اگر به اسم rostopic را بزنیم یک سری کامندها و توضیحات آن‌ها را معرفی میکند. اگر اینجا هم rostopic list را بزنیم دو تاپیک دیفالت را فقط نشان میدهد. پس فعلا نه نود خاصی و نه topic خاصی فعلا داریم.

حال میخواهیم یک نود را ران کنیم برای این کار باید از دستور rosrun استفاده کنیم و بعد اسم پکیج را بگوییم. در اینجا از پکیج آماده turtlesim استفاده میکنیم. سپس بعد از آن باید اسم نود مورد نظر که turtlesim\_node میباشد می آید.



A screenshot of a computer

Description automatically generated with medium confidenceبعد از اجرای آن صفحه‌ی GUI زیر باز میشود.

حال یک ترمینال جدید باز میکنیم و مجدد لیست نودها و تاپیک‌ها را مشاهده میکنیم. یک نود turtlesim میبینیم که اضافه شده و همچنین 3 تا تاپیک جدید نیز اضافه شده است.

Text

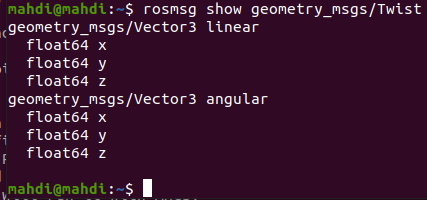
Description automatically generated

مثلا در تاپیک cmd\_vel قراره سرعت حرکات ربات قرار بگیرد. باید توجه داشت که هر تاپیک massage type مخصوص خودش را دارد. وقتی یک نود publisher درست میکنید اون تاپیک شما یک مسیج تایپی دارد و بعدا اگر نودی بخواهد از آن تاپیک subscribe کند باید حتما از آن تایپ استفاده کند و اگرنه نمیتواند از تاپیک استفاده کند. پس مثلا مهمه که تایپ مسیجی که در تاپیک cmd\_vel یا تاپیک های دیگه هست را بفهمیم. برای این کار باید اطلاعات اضافه تری درباره خود آن تاپیک داشته باشیم ولی الان چیزی که از تاپیک‌ها میدونیم فقط اسمشان هست. به کمک rostopic info میتوان اطلاعات تاپیک active Text

Description automatically generatedرا به دست آورد.

همانطور که مشاهده میشود تایپ مسیج آن geometry\_msgs/Twist میباشد. پابلیشری ندارد و کسی هم که subscribe میکند همان turtlesim میباشد. درواقع نود ران شده برای اینکه بتواند ربات را تکان دهد باید بتواند این تاپیک را subscribe کند و مسیج‌های سرعت را بگیرد و ربات را تکون بده و خب الان پابلیشر نداره و ربات تکانی نمیخورد.

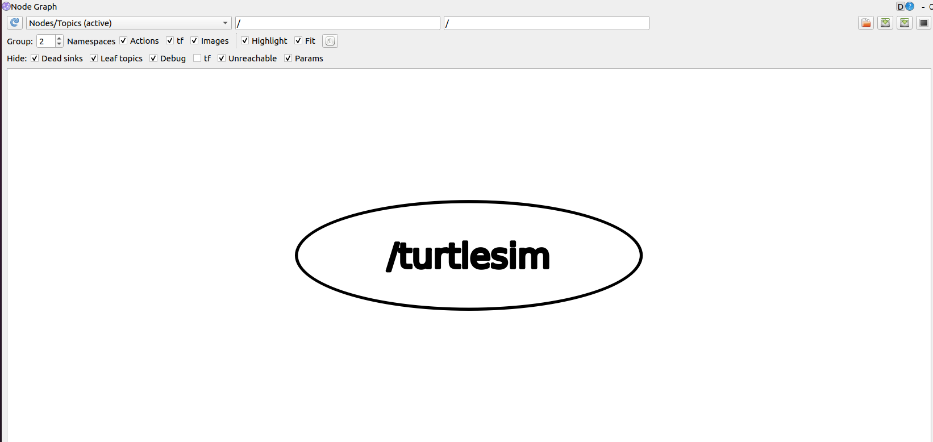
حال اگر بخواهیم راجع به خود مسیج بیشتر بدانیم میتوان به صورت زیر عمل کرد. همانطور که در زیر میبینید میگوید Twist از دوتا وکتور تشکیل شده که یکیش linear و دیگری angular میباشد. که linear سرعت خطی را مشخص میکند ( در 3 راستا) و angular هم سرعت‌های زاویه‌ای حول yaw و pitch و roll میباشد.



ابزار دیگری که وجود دارد و میتوان با آن کار کرد rqt\_graph میباشد.



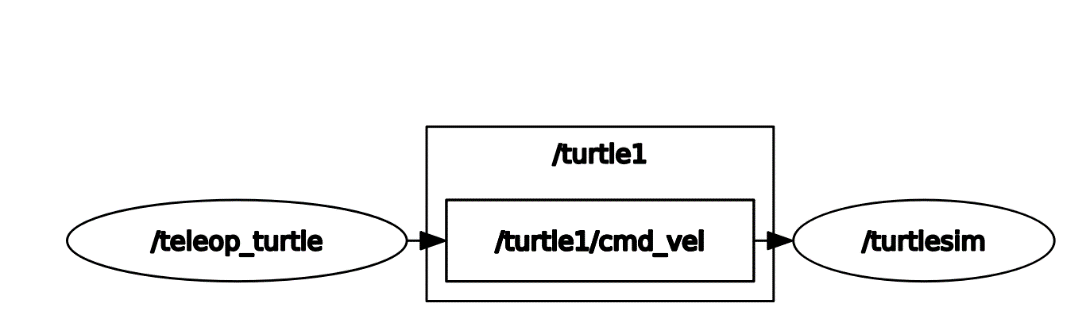
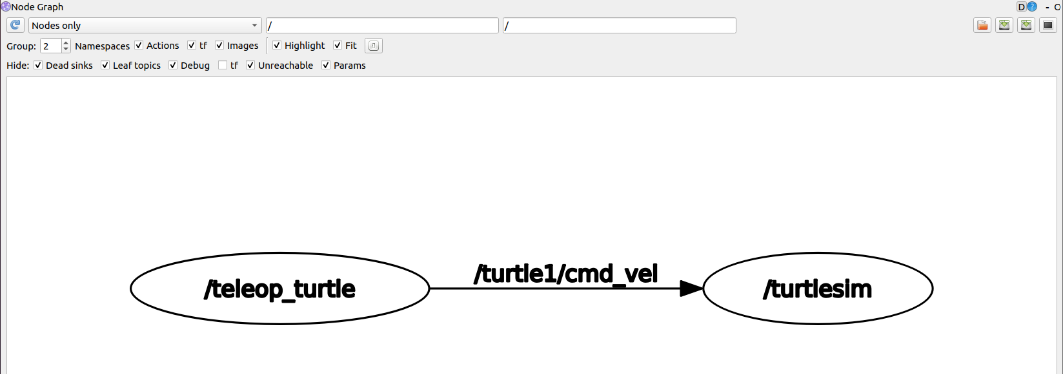
در صورت اجرای آن صفحه ای به صورت زیر باز میشود:



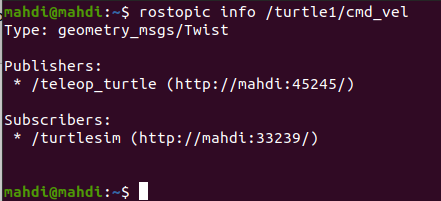
Graphical user interface, text

Description automatically generatedاون بیضی بالا درواقع نودی هست که داریم و نود دیگه‌ای ران نشده و چون کسی نیست که این تاپیکی که نود turtlesim میخواهد را publish کند و اطلاعاتش را پر کند پس اون تاپیک را هم نشان نمیدهد. حال یک نود اضافه کنیم به گونه‌ای که بتواند ربات را حرکت دهد. یک ترمینال جدید باز کرده و در آن به کمک rosrun نود مربوطه را از همین پکیج turtlesim ران میکنیم. این نود کاری که میکند این است که اینتراپت‌هایی که روی ترمینال از کیبورد می افتد را میخواند و اگر از arrow keyهای کیبورد استفاده کرده باشید میتوانید باهاش ربات را حرکت دهید. که برای امتحان میتوانید از کیبورد خود برای حرکت ربات استفاده کنید و خروجی را ببینید:

حال اگر rqt\_graph را رفرش کنیم تصویر زیر را خواهیم دید. درواقع نود جدیدی که ران کردیم تاپیک cmd\_vel را publish میکند و نود turtlesim آن را subscribe میکند. میتوان شکل نمایش را تغییر داد در گراف به صورتی که تاپیک را داخل مستطیل نشان دهد.



دستور rostopic info که قبلا ران کردیم میگفت این تاپیک cmd\_vel هیچ publisherای ندارد ولی اگر الان مجدد اجرا کنیم میگوید teleop\_turtle پابلیشر میباشد.

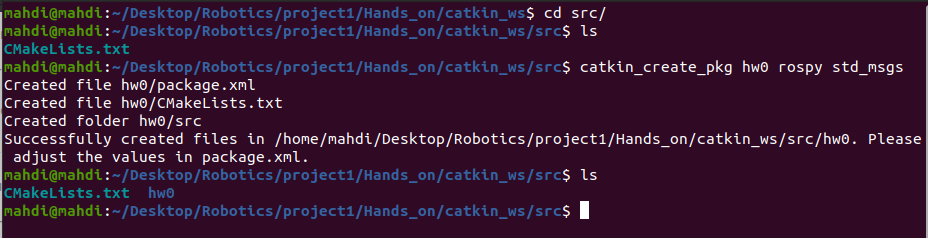


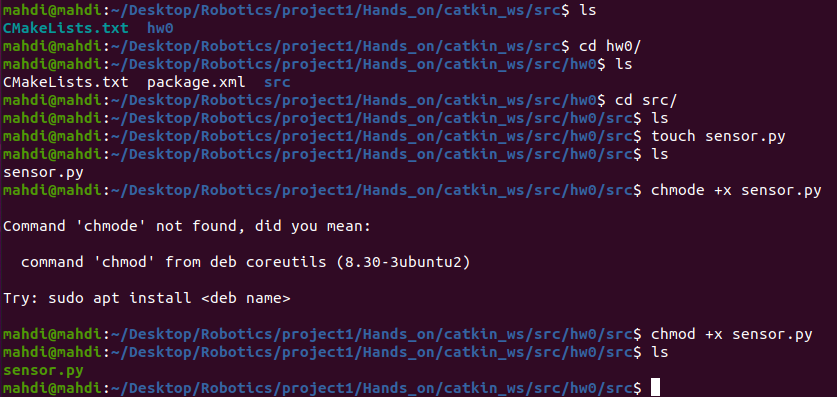
همچنین اگر لیست نودها را باز پرینت کنیم میبنیم تعدادی نود اضافه شده است:

Text

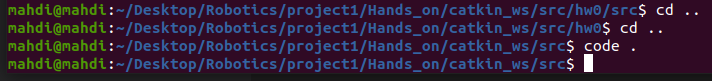
Description automatically generated

تا اینجای کار یک دید کلی نسبت به نود و تاپیک به دست اوردیم. هر نود صرفا یک برنامه قابل اجراس که یک هدف خاصی دارد و اون هدفه میتواند بالا اوردن یک GUI باشد یا کار با یک سنسور یا کنترلر باشد. تاپیک هم یک اینترفیسی بین نودهای مختلف برای آنکه massage منتقل کنند، میباشد. مثل دیوار مهربانی میباشد و هر تاپیک یه سری مسیج ها روش مینشینند و هر نودی بخواد میاد اون مسیج ها را برمیدارد.

حال نودهای باز شده و ترمینال‌ها را میبندیم. به جز ترمینال مربوط به roscoreکه در آن هم کنترل+C زده تا مستر خاموش شود. حال به پوشه src رفته که در آنجا یک CMakeLists.txt داریم. در این پوشه سورس باید پکیج ایجاد کنیم. دستور آن catkin\_create\_pkg name میباشد که name اسم پکیج و دلخواه میباشد. مثلا ما میگذاریم hw0 و بعد از آن یک سری dependency را add میکنیم مثلا میدانیم در این پکیج باید از rospy و std\_msgs استفاده کنیم.

داخل این پکیج ساخته شده میرویم و اگر ls بزنیم 3 تا اسم نمایش میدهد. ازمیان آن‌ها ما کدهامون را داخل فولدر src قرار میدهیم. برای نوشتن کد داخل فولدر src میرویم که در آن هم خالی میباشد. اول یک فایل درست میکنیم. طبق دستور کار، یک نودی که ابتدا درست میکنیم نود سنسور میباشد، پس اینجا هم اسم فایل پایتون را سنسور میگذاریم. این فایل executable نیست برای این که قابل اجرا شود از chmod استفاده میکنیم.

مراحل بعدی را از داخل vscode انجام میدهیم. در ترمینال بک میزنیم تا به پوشه src اول برسیم.

(دقت داشته باشید که اکستنشن python از مایکروسافت و اکستنشن ROS از مایکروسافت و اکستنشن CMake از twxs را روی vscode نصب داشته باشید)

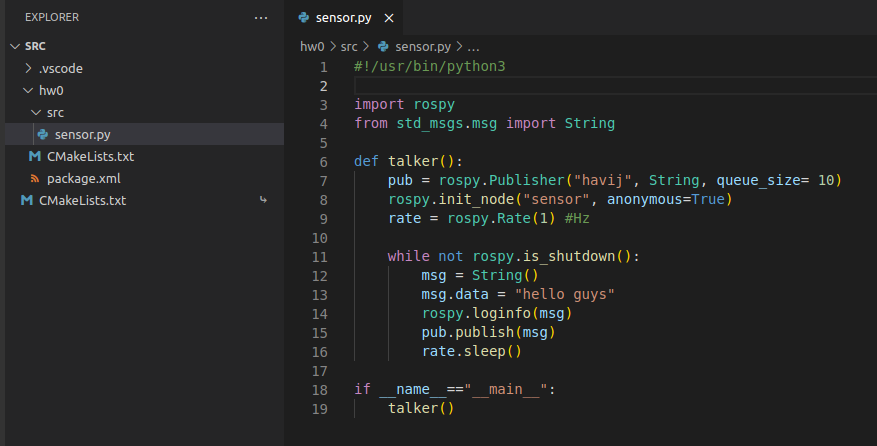
سپس فایلsensor.py را باز میکینم. در آن میخواهیم یک نود خیلی ساده درست کنیم. یک خط کد ابتدا باید بنویسیم که در حقیقت interpreter code میباشد و باعث میشود که موقع اجرای فایل متوجه شود که مفسری که باید استفاده شود پایتون3 میباشد.

#!/usr/bin/python3

سپس باید کتابخانه rospy را import کنیم. زمانی که بخوایم publisher یا نود داخل کد درست کنیم حتما باید از این لایبری باید استفاده نماییم. میخواهیم حالا صرفا یک نودی درست کنیم که یک پیام Hello world ساده بنویسه.

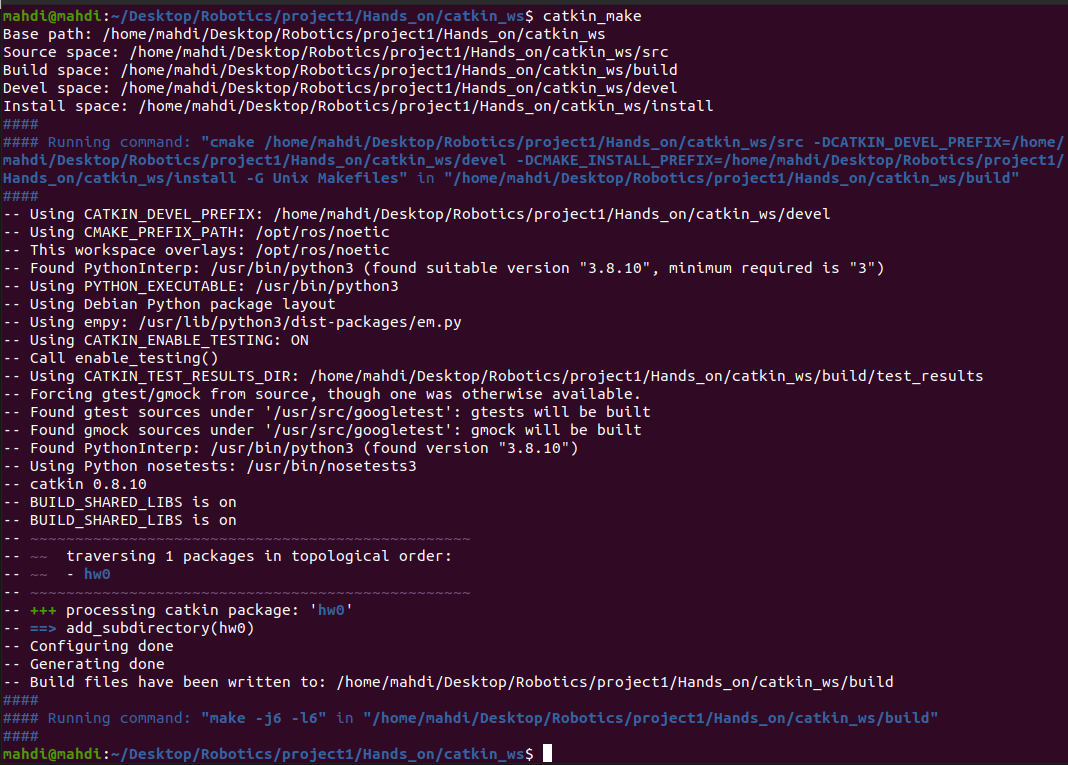
در فانکشن talker ابتدا یک publisher لازم است که تعریف شود. داخل آن باید اسم تاپیک و بعد تایپ مسیج که میخواهیم بذاریم و همچنین سایز بافر(queue size) را مشخص کنیم. (در این جا از تایپ String میخوایم استفاده کنیم که نمیشناسد پس لازم است تا from std\_msgs.msg import String را به ابتدای کد اضافه کنیم.) درباره اون بافر سایز میتوان گفت اگر شبکه کند بود و دستوراتی که قراره توی تاپیک ما پابلیش بشه نمیرفت توی اون بافر نگه میداره که دستورها ازبین نروند و هروقت مناسب بود پابلیش میکنه از اونجا. سپس لازم است تا خود node را initialize کنیم که در تعریف ان صرفا یک اسم میگذاریم که این اسمی هست که برای نود میخوایم مثلا در اینجا sensor میگذاریم. یک متغیر دیگری به اسم anonymous در این متد هست که میتوانید آن را True یا False بگذارید. (دیفالت False هست). توی ROS ممکنه چندتا فایل کد داشته باشید و توی هرکدوم یک initialize node انجام داده باشید و اگر اشتباها اسم چندتا از این نودها را یکی گذاشته باشید ، اون نودی که اخر از همه اجرا میشود بقیه را over write میکند و نودهای دیگه اجرا نمیشن که برای حل این مشکل میتوان anonymous=true گذاشت و یک مقدارعدد رندومی به ته اون کلمه سنسور اضافه میکند که مطمئن باشد فقط یدونه از اون نود هست. پس خوبه ازش استفاده کنید.

سپس یک لوپ درست میکنیم که در آن کارهایی که میخوایم انجام دهیم را به صورت continues انجام دهیم. این لوپ بنهایت است اما یک شرط خروج براش میذاریم که اگر یک اینتراپت دریافت کرد خارج شود. ( not rospy.is\_shutdown() )

حال در داخل لوپ ابتدا یک متغیر مسیج از جنس String تعریف میکنیم و سپس مقدار data را در آن تعیین میکنیم. برای اینکه داخل ترمینال یک log بندازه و پرینت کنه ببینیم چه خبره میتوان از rospy.loginfo(“”) استفاده کرد که همون کار print() را برای ما انجام میدهد. سپس لازم است که کار publish را به کمک متغیر publisherای که تعریف کرده بودیم انجام دهیم.

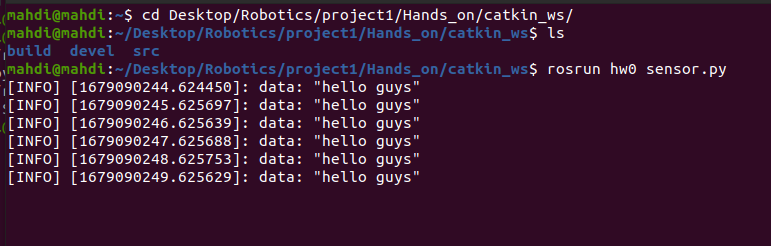
فقط این کار داخل while الان داره با سرعت cpu انجام میشود و ممکن است بخواهیم هر یک ثانیه یک بار پابلیش شود. برای این امر بیرون از while یک بار rate را تعریف میکنیم که در آن ورودی عددی برحسب فرکانس ( واحدش هرتزه) میباشد و مثلا 10 یعنی در هرثانیه 10 بار . بعد داخل لوپ اخرش هم مینویسیم rate.sleep()

حال فایل کد را save میکنیم. چون تغییرات دادیم باید یک بار دیگر catkin\_make را در داخل پوشه catkin\_ws انجام دهیم.

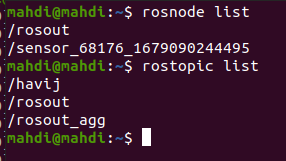


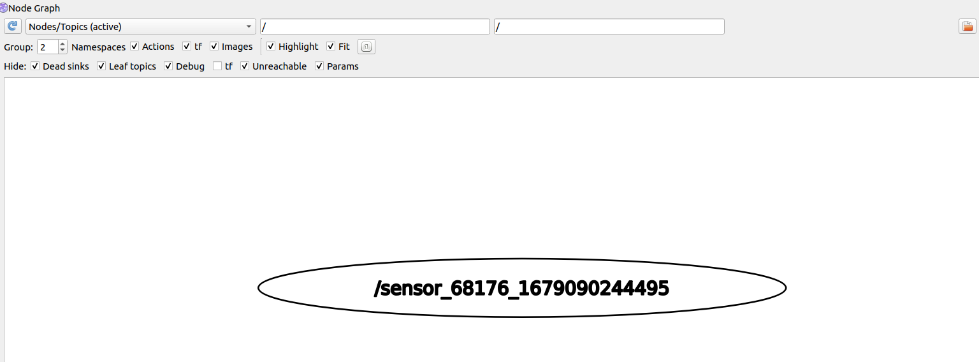
حال roscore را در همین ترمینال ران میکنیم. سپس یک ترمینال جدید نیز باز کرده و وارد work space میشویم. سپس اگه بخوایم بگیم rosrun hw0 نمیشناسه، دلیلش اینه setup.bash که داخل فولدر devel هست را source نکردم. پس مینویسیم :

. devel/setup.bash

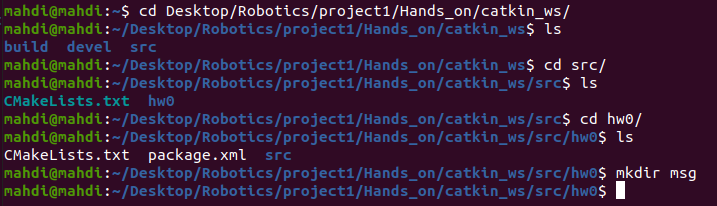
حال اگر موارد گفته شده را انجام دهیم و نود را اجرا کنیم:

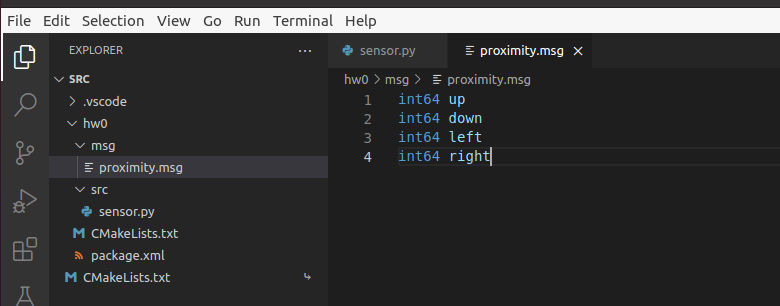
الان برای اینکه مطمئن شویم خود topic به درستی کار میکند یک rosnode list میزنیم و نود سنسور را میبینیم و همچنین در لیست تاپیک‌ها، تاپیک هویج را نیز مشاهده خواهیم کرد:



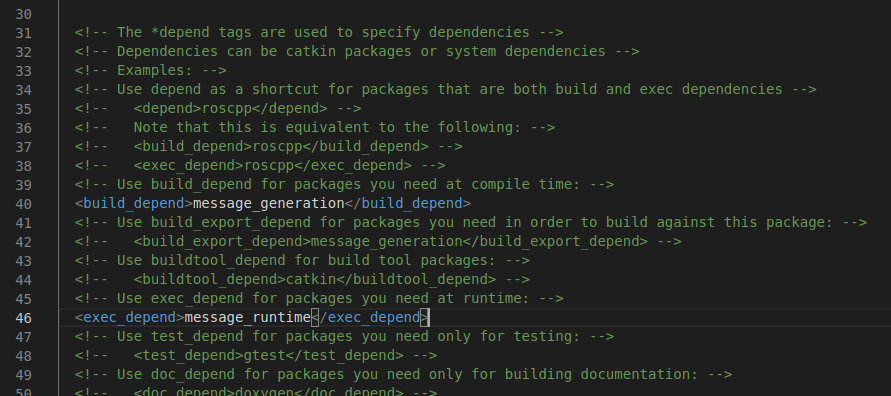
همچنین اگر rqt\_graph هم بزنیم نود را خواهیم دید ولی خب چون از تاپیک کسی subscribe نکرده آن را نخواهیم دید.

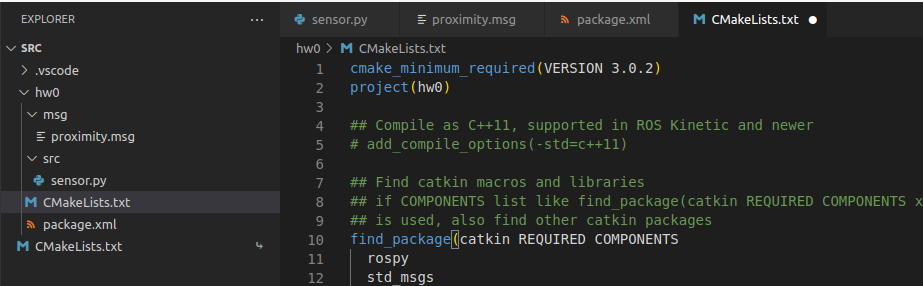
حال باتوجه به صورت تمرین میخواهیم کمی کد را modify کنیم و تا جای ممکن نزدیک به نودی شود که برای تمرین صفر نیاز است.

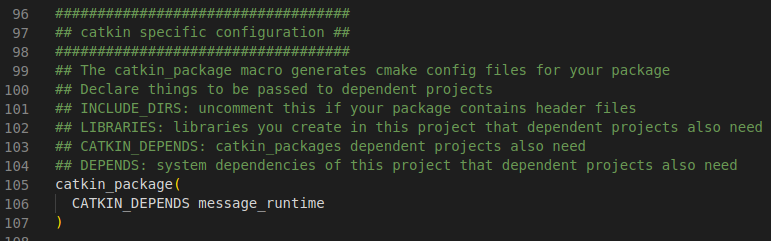
اسم تاپیک را distance میگذاریم. نوع مسیج 4 تا integer میباشد و string به درد ما نمیخورد. حالا رندوم عدد ایجاد کردن را نگفت ولی فعلا میخوایم custom message ایجاد کنیم. برای این کار وارد پوشه src میشویم و سپس وارد پکیج شده که در انجا یک فولدر src داریم. یک فولدر دیگه به اسم msg ایجاد میکنیم. همه فایلای مسیج را قراره در این فولدر اضافه کنیم.

حال در vscode روی msg کلیک راست کرده و new file میزنیم و چون قراره اعداد سنسور در آن قرار بگیره اسمش رو proximity.msg میگذاریم. حال در این فایل باید مسیجم را تعریف کنم. من 4 تا مقدار عددی دارم و اعدادم هم integer اند.

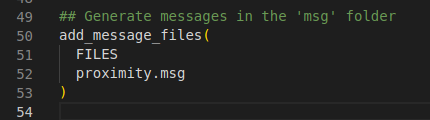
برای اینکه این مسیج را توی فایل کد بشناسه، باید تغییراتی را داخل CMakeLists.txt و package.xml بدهیم. برای این منظور میتوانید به قسمت tutorials سطح beginner در wiki.ros مراجعه کنید و قسمت CreatingMsgAndSrv بروید. ([لینک](http://wiki.ros.org/ROS/Tutorials/CreatingMsgAndSrv)) براساس آن قرار است پیش برویم.

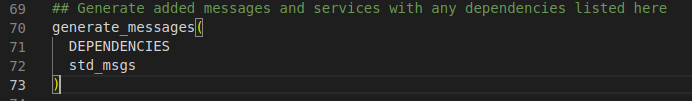
اولین تغییر در package.xml میباشد. دو خطی که در خطوط 40 و 46 هستند را uncomment کنید.

حال باید به سراغ تغییرات داخل CMakeLists ( اونی که داخل پکیج hw0 هست) برویم. یکی اینکه در find\_package در خط 10 باید message\_generation را اضافه نماییم.

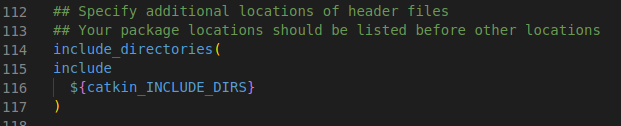
تغییر دیگر در catkin\_package در خط 105 میباشد:

سپس به دنبال add\_message\_files میگردیم و آن را uncomment میکنیم و proximity.msg را به جای Message1.msg و Message2.msg میگذاریم.



حال باید generate\_message را از comment در بیاریم.

کار دیگری که باید بکنیم اینه که در خط 115 اون include را هم uncomment کنیم.

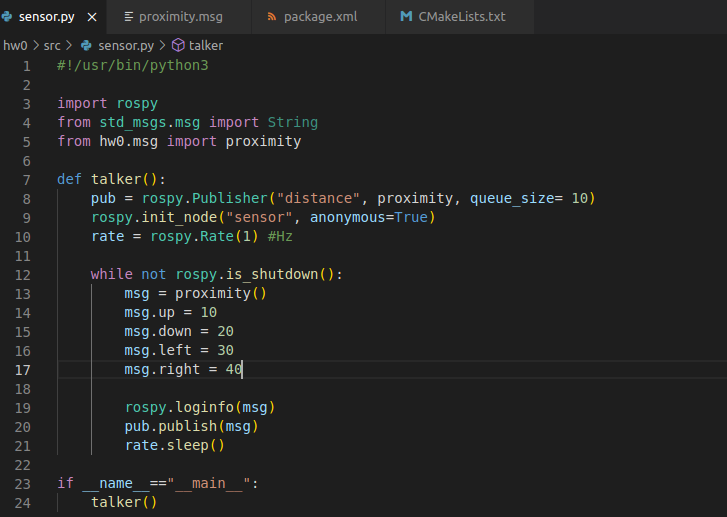


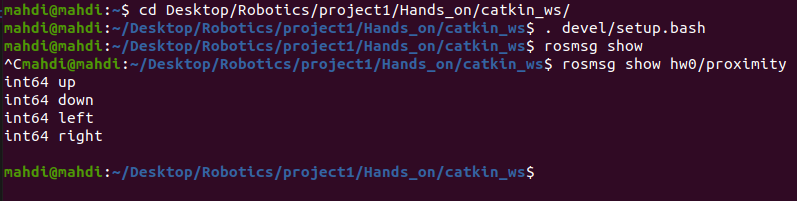
چون دوباره پکیج جدید درست کردیم باید catkin\_make را صدا بکنیم.(در همان ترمینالی که rescore فعال بود ابتدا با Ctrl+C آن را ازکار انداخته و بعد دستور را میزنیم.)

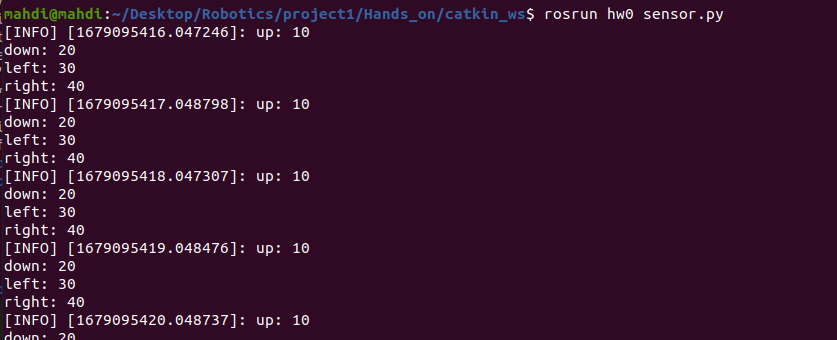
حال یک بار هم vscode را بسته و دوباره باز میکنیم. حال میتوان import را انجام داد.

from hw0.msg import proximity

همچنین داخل متد publisher هم تایپ را proximity میگذاریم. در گام بعد باید messageهایی که داخل تاپیک میگذاریم را داخل لوپ عوض کنیم. یک مسیج از جنس proximity درست میکنیم و بعد مادیر متغیرهای up، down و ... را ست میکنیم.



حال باید امتحان کنیم ببینیم به درستی کار میکند یا نه، برای این کار roscore را صدا میکنیم. در یک ترمینال دیگر وارد catkin\_ws شده، سپس چک میکنیم آیا مسیج ما هست یا نه. برای این کار rosmsg show را میزنیم که میبینیم hw0/ وجود دارد. سپس با نمایش مسیج تاپیک proximity خواهیم دید:

حال میتوان نود موردنظر را اجرا کرد:

# سوال 3 – عنوان سوال