



Department of
Computer Engineering

به نام خدا



Amirkabir University of Technology
(Tehran Polytechnic)

دانشگاه صنعتی امیرکبیر
دانشکده مهندسی کامپیوتر
اصول علم ربات

تمرين سري اول

نام و نام خانوادگی	مهندی رحمانی
شماره دانشجویی	۹۷۳۱۷۰۱
تاریخ ارسال گزارش	۱۴۰۱/۱۲/۲۷

فهرست گزارش سوالات

۳.....	بخش ۱ – نصب نرم افزار
۵.....	بخش ۲ – انجام تمرین هندزان از روی ویدئو.....
۲۰	بخش ۳ – مراحل ساختن نودهای خواسته شده در دستورکار
۳۱	بخش ۴ – صحت سنجی نودها و اسکرین شات از آنها.....

بخش ۱ - نصب نرم افزار

ابتدا با توجه به ویدیویی قرار داده شده، سعی میکنیم تا ROS را نصب کنیم. با توجه به اینکه نسخه ROS مورد استفاده ROS-noetic، 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:

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

سپس در گام بعد با توجه 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`

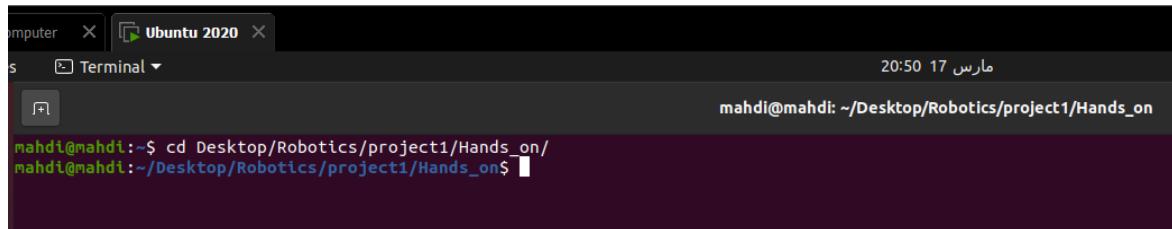
➤ Initialize rosdep

- `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.

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



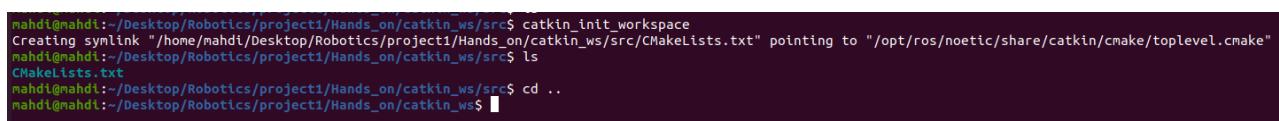
```
mahdi@mahdi:~$ cd Desktop/Robotics/project1/Hands_on/
```

سپس 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 فolder شویم میبینیم یک فolderی به اسم src ساخته شده که ولی داخل آن فعلا چیزی وجود ندارد.



```
mahdi@mahdi:~$ cd Desktop/Robotics/project1/Hands_on/
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on$ mkdir -p catkin_ws/src
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on$ cd catkin_ws/
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws$ ls
src
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws$ cd src/
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src$ ls
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src$
```

لازم است تا در این فolder initialize work space را بکنیم. دستور مربوطه میباشد. سپس ls میزنیم و میبینیم که cd ایجاد شده است. در نهایت .. میزنیم که به پوشه قبلی برگردیم.



```
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src$ catkin_init_workspace
Creating symlink "/home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/src/CMakeLists.txt" pointing to "/opt/ros/noetic/share/catkin/cmake/toplevel.cmake"
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src$ ls
CMakeLists.txt
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src$ cd ..
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws$
```

حال باید دستور catkin_make را اجرا کنیم. و بعد اگر ls را بزنیم خواهیم دید دو فolder build و devel اضافه شدند. ما همهی کدها و نودها و چیزهایی که میسازیم را در پوشه src قرار میدهیم. این building tool یک src میباشد و چیزهایی که توی src هست مثل نودها را در صورت لزوم در

فولدرهای build و devel قرار میدهد. اگر مثلاً 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 کار کنیم لازم است آن را source کنیم.

```
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws$ catkin_make
Base path: /home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws
Source space: /home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/src
Build space: /home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/build
Devel space: /home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/devel
Install space: /home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/install
#####
##### Running command: "cmake /home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/src -DCATKIN_DEVEL_PREFIX=/home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/devel -DCMAKE_INSTALL_PREFIX=/home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/install -G Unix Makefiles" in "/home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/build"
#####
-- The C compiler identification is GNU 9.4.0
-- The CXX compiler identification is GNU 9.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Checked for working CXX compiler: /usr/bin/c++
-- Checked for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Using CATKIN_DEVEL_PREFIX: /home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/devel
-- Using CMAKE_PREFIX_PATH: /opt/ros/noetic
-- This workspace overlays: /opt/ros/noetic
-- Found PythonInterp: /usr/bin/python3 (found suitable version "3.8.10", minimum required is "3")
-- Using PYTHON_EXECUTABLE: /usr/bin/python3
-- Using Debian Python package layout
-- Found PY_om: /usr/lib/python3/dist-packages/en.py
-- Using empty: /usr/lib/python3/dist-packages/en.py
-- Using CATKIN_ENABLE_TESTING: ON
-- Call enable_testing()
-- Using CATKIN_TEST_RESULTS_DIR: /home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/build/test_results
-- Forcing gtest/gmock from source, though one was otherwise available.
-- Found gtest sources under '/usr/src/googletest': gtests will be built
-- Found gmock sources under '/usr/src/googletest': gmock will be built
-- Found PythonInterp: /usr/bin/python3 (found version "3.8.10")
-- Found Threads: TRUE
-- Using Python nosetests: /usr/bin/nosetests3
-- catkin 0.8.10
-- BUILD_SHARED_LIBS is on
-- BUILD_SHARED_LIBS is on
-- Configuring done
-- Generating done
-- Build files have been written to: /home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/build
#####
##### Running command: "make -j6 -l6" in "/home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/build"
#####
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws$ ls
build devel src
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws$ cd devel/
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/devel$ ls
cmake.lock env.sh lib local_setup.bash local_setup.zsh local_setup.zsh setup.bash setup.sh _setup_util.py setup.zsh
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/devel$ ..
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws$ 
```

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 Package ها و نودها وجود دارند که واسه هاش هستند و میتوانید از آنها استفاده کنید. ماهم در ادامه از اینا استفاده میکنیم و ران میکنیم Tutorial تا با نود و تاپیک اشنا شویم. برای این منظور در همین ترمینال roscore را اجرا کنیم:

```
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws$ roscore
... logging to /home/mahdi/.ros/log/7d89c32c-c4eb-11ed-834a-511777696eca/roslaunch-mahdi-63047.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://mahdi:46577/
ros_comm version 1.15.15

SUMMARY
========
PARAMETERS
  * /rosdistro: noetic
  * /rosversion: 1.15.15

NODES

auto-starting new master
process[master]: started with pid [63058]
ROS_MASTER_URI=http://mahdi:11311/

setting /run_id to 7d89c32c-c4eb-11ed-834a-511777696eca
process[rosout-1]: started with pid [63068]
started core service [/rosout]
```

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 مربوط به آنها را میتوان دید.

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

```
mahdi@mahdi:~$ rosnode
rosnode is a command-line tool for printing information about ROS Nodes.

Commands:
    rosnode ping      test connectivity to node
    rosnode list      list active nodes
    rosnode info      print information about node
    rosnode machine   list nodes running on a particular machine or list machines
    rosnode kill       kill a running node
    rosnode cleanup   purge registration information of unreachable nodes

Type rosnode <command> -h for more detailed usage, e.g. 'rosnode ping -h'

mahdi@mahdi:~$ rosnode list
/rosout
mahdi@mahdi:~$
```

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 فعلاً نه نود خاصی و نه topic خاصی فعلاً داریم.

```
mahdi@mahdi:~$ rostopic
rostopic is a command-line tool for printing information about ROS Topics.

Commands:
    rostopic bw        display bandwidth used by topic
    rostopic delay     display delay of topic from timestamp in header
    rostopic echo      print messages to screen
    rostopic find      find topics by type
    rostopic hz        display publishing rate of topic
    rostopic info      print information about active topic
    rostopic list      list active topics
    rostopic pub       publish data to topic
    rostopic type      print topic or field type

Type rostopic <command> -h for more detailed usage, e.g. 'rostopic echo -h'

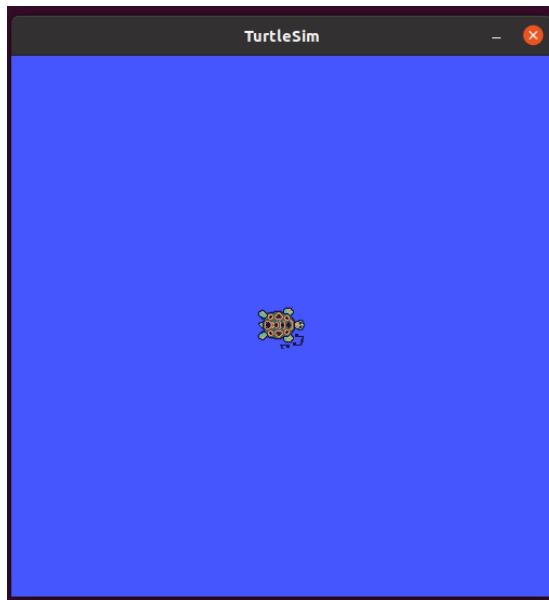
mahdi@mahdi:~$ rostopic list
/rosout
/rosout_agg
mahdi@mahdi:~$
```

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.

```
mahdi@mahdi:~$ rosrun turtlesim turtlesim_node
[ INFO] [1679080568.172760976]: Starting turtlesim with node name /turtlesim
[ INFO] [1679080568.174983360]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]

```

بعد از اجرای آن صفحه‌ی GUI زیر باز می‌شود.



حال یک ترمینال جدید باز میکنیم و مجدد لیست نودها و تاپیک‌ها را مشاهده میکنیم. یک نو德 turtlesim میبینیم که اضافه شده و همچنین ۳ تا تایک جدید نیز اضافه شده است.

```
mahdi@mahdi:~$ rosnode list  
/rosout  
/turtlesim  
mahdi@mahdi:~$ rostopic list  
/rosout  
/rosout_agg  
/turtle1/cmd_vel  
/turtle1/color_sensor  
/turtle1/pose  
mahdi@mahdi:~$
```

مثلا در تاپیک cmd_vel قراره سرعت حرکات ربات قرار بگیرد. باید توجه داشت که هر تاپیک massage مخصوص خودش را دارد. وقتی یک نود publisher درست میکنید اون تاپیک شما یک مسیج type 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 مثلاً 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 را به دست آورد.

```
mahdi@mahdi:~$ rostopic info /turtle1/cmd_vel
Type: geometry_msgs/Twist

Publishers: None

Subscribers:
* /turtlesim (http://mahdi:33239/)

mahdi@mahdi:~$
```

همانطور که مشاهده میشود تایپ مسیج آن geometry_msgs/Twist میباشد. پابلیشری ندارد و کسی هم که 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 ۳ راستا) و angular هم سرعت‌های زاویه‌ای حول yaw و roll و pitch میباشد.

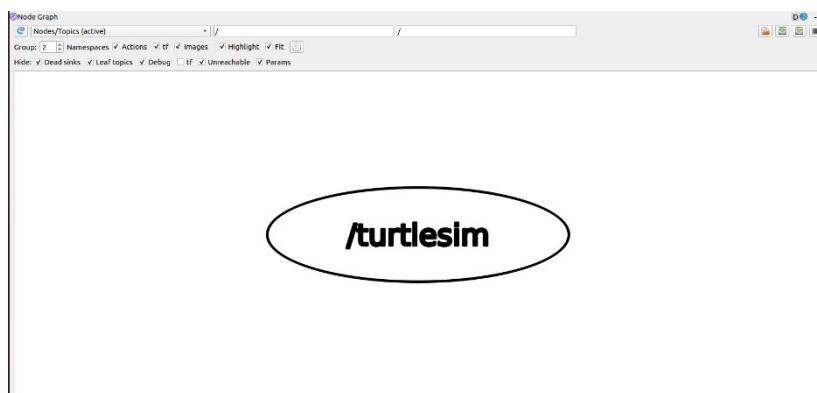
```
mahdi@mahdi:~$ rosmsg show geometry_msgs/Twist
geometry_msgs/Vector3 linear
  float64 x
  float64 y
  float64 z
geometry_msgs/Vector3 angular
  float64 x
  float64 y
  float64 z

mahdi@mahdi:~$
```

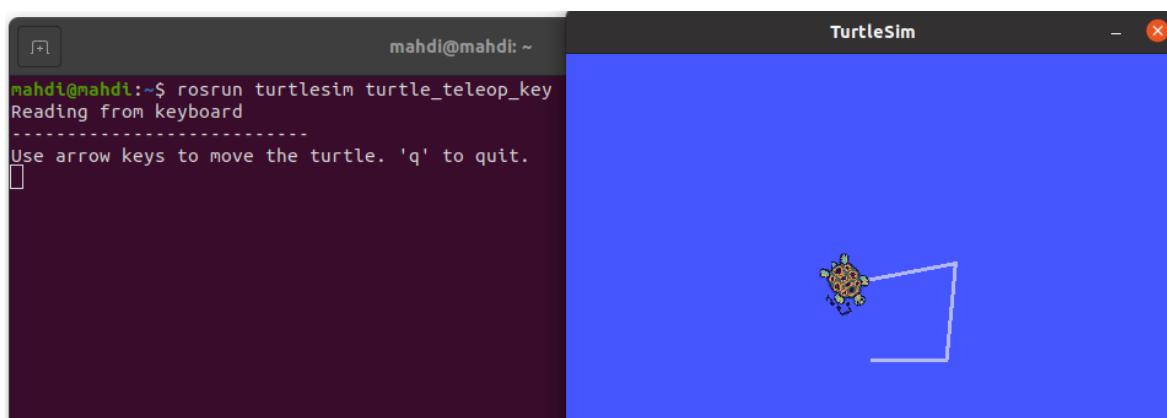
ابزار دیگری که وجود دارد و میتوان با آن کار کرد rqt_graph میباشد.

```
mahdi@mahdi:~$ rqt_graph
```

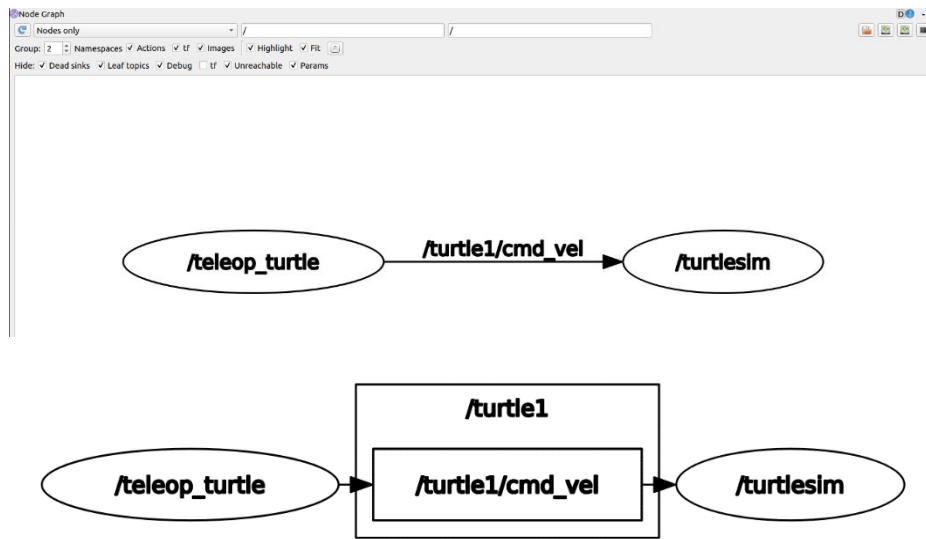
در صورت اجرای آن صفحه ای به صورت زیر باز میشود:



اون 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 اینترپاتهایی که روی ترمینال از کیبورد می‌افتد را میخواند و اگر از key arrow 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 turtlesim را subscribe میکند و نود cmd_vel را publish میکند. میتوان شکل نمایش را تغییر داد در گراف به صورتی که تاپیک را داخل مستطیل نشان دهد.



دستور rostopic info که قبل ران کردیم میگفت این تاپیک cmd_vel هیچ publisher ندارد ولی اگر الان مجدد اجرا کنیم میگوید teleop_turtle پابلیشر میباشد.

```
mahdi@mahdi:~$ rostopic info /turtle1/cmd_vel
Type: geometry_msgs/Twist

Publishers:
* /teleop_turtle (http://mahdi:45245/)

Subscribers:
* /turtlesim (http://mahdi:33239/)

mahdi@mahdi:~$
```

همچنین اگر لیست نودها را باز پرینت کنیم میبینیم تعدادی نod اضافه شده است:

```
mahdi@mahdi:~$ rosnode list
/rosout
/rqt_gui_py_node_63695
/teleop_turtle
/turtlesim
mahdi@mahdi:~$
```

تا اینجای کار یک دید کلی نسبت به نود و تاپیک به دست اوردیم. هر نود صرفاً 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 مسیح ها روش مینشینند و هر نودی بخود میاد اون مسیح ها را برمیدارد.

حال نودهای باز شده و ترمینال ها را میبینیدم. به جز ترمینال مربوط به که در آن هم کنترل C+ roscore 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. مثلاً ما میگذاریم hw0 و بعد از آن یک سری add dependency را میکنیم مثلاً میدانیم در این پکیج باید از rospy و std_msgs استفاده کنیم.

```
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws$ cd src/
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src$ ls
CMakeLists.txt
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src$ catkin_create_pkg hw0 rospy std_msgs
Created file hw0/package.xml
Created file hw0/CMakeLists.txt
Created folder hw0/src
Successfully created files in /home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/src/hw0. Please
adjust the values in package.xml.
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src$ ls
CMakeLists.txt  hw0
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src$ 
```

داخل این پکیج ساخته شده میرویم و اگر ls بزنیم ۳ تا اسم نمایش میدهد. از میان آنها ما کدهامون را داخل فolder src قرار میدهیم. برای نوشتن کد داخل فolder src میرویم که در آن هم خالی میباشد. اول یک فایل درست میکنیم. طبق دستور 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.

```
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src$ ls
CMakeLists.txt  hw0
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src$ cd hw0/
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src/hw0$ ls
CMakeLists.txt  package.xml  src
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src/hw0$ cd src/
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src/src/hw0$ ls
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src/src/hw0$ touch sensor.py
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src/src/hw0$ ls
sensor.py
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src/hw0/src$ chmod +x sensor.py

Command 'chmod' not found, did you mean:

  command 'chmod' from deb coreutils (8.30-3ubuntu2)

Try: sudo apt install <deb name>

mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src/hw0/src$ chmod +x sensor.py
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src/hw0/src$ ls
sensor.py
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src/hw0/src$ 
```

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

```
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src/hw0/src$ cd ..  
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src/hw0$ cd ..  
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src$ code .  
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src$ █
```

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

سپس فایل sensor.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 پایتون ۳ میباشد.

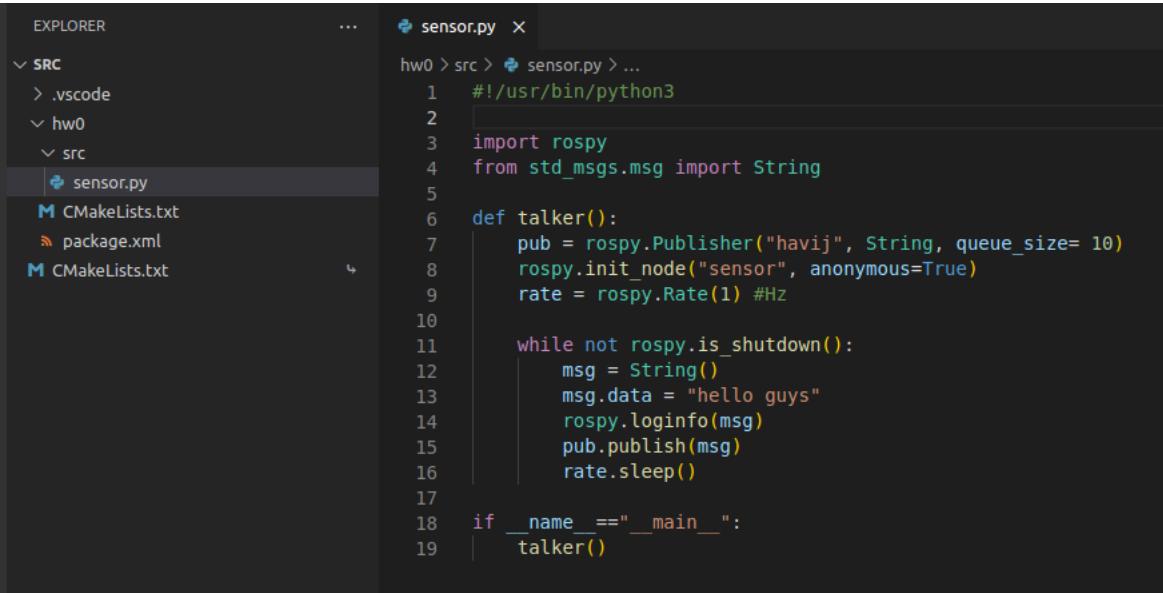
```
#!/usr/bin/python3
```

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 اینجا از تایپ String میخوایم استفاده کنیم که نمیشناسد پس لازم است تا from std_msgs.msg import String کنیم که در تعريف ان صرفا یک 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 از بین نرونده هر وقت مناسب بود پابلیش میکنه از اونجا. سپس لازم است تا خود node را initialize کنیم که در تعريف ان صرفا یک اسم میگذاریم که این اسمی هست که برای نود میخوایم مثلًا در اینجا sensor میگذاریم. یک متغیر دیگری به اسم anonymous در این متده است که میتوانید آن را True یا False بگذارید. (دیفالت هست). توی ROS ممکنه چندتا فایل کد داشته باشید و توی هر کدام یک initialize node را انجام داده باشید و اگر اشتباهها 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 مقدار عدد 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 بنهاشت است اما یک شرط خروج برash میذاریم که اگر یک اینتراتپ 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 print() استفاده کرد که همون کار rospy.loginfo("") را برای ما انجام میدهد. سپس لازم است که کار publisher را به کمک متغیر `pub` ای که تعریف کرده بودیم انجام دهیم.



The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays a project structure with a 'SRC' folder containing '.vscode', 'hw0', and 'src'. Inside 'src', there is a 'sensor.py' file which is currently selected and shown in the main editor area. The code in 'sensor.py' is as follows:

```

EXPLORER ... sensor.py ...
SRC
> .vscode
hw0
> src
| > sensor.py
M CMakeLists.txt
package.xml
M CMakeLists.txt

sensor.py
1 #!/usr/bin/python3
2
3 import rospy
4 from std_msgs.msg import String
5
6 def talker():
7     pub = rospy.Publisher("havij", String, queue_size= 10)
8     rospy.init_node("sensor", anonymous=True)
9     rate = rospy.Rate(1) #Hz
10
11    while not rospy.is_shutdown():
12        msg = String()
13        msg.data = "hello guys"
14        rospy.loginfo(msg)
15        pub.publish(msg)
16        rate.sleep()
17
18 if __name__=="__main__":
19     talker()

```

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 ۱۰ یعنی در هر ثانیه ۱۰ بار . بعد داخل لوپ اخres 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.

```

mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws$ catkin_make
Base path: /home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws
Source space: /home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/src
Build space: /home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/build
Devel space: /home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/devel
Install space: /home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/install
#####
##### Running command: "cmake /home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/src -DCATKIN_DEVEL_PREFIX=/home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/devel -DCMAKE_INSTALL_PREFIX=/home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/install -G Unix Makefiles" in "/home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/build"
#####
-- Using CATKIN_DEVEL_PREFIX: /home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/devel
-- Using CMAKE_PREFIX_PATH: /opt/ros/noetic
-- This workspace overlays: /opt/ros/noetic
-- Found PythonInterp: /usr/bin/python3 (found suitable version "3.8.10", minimum required is "3")
-- Using PYTHON_EXECUTABLE: /usr/bin/python3
-- Using Debian Python package layout
-- Using empy: /usr/lib/python3/dist-packages/empy.py
-- Using CATKIN_ENABLE_TESTING: ON
-- Call enable_testing()
-- Using CATKIN_TEST_RESULTS_DIR: /home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/build/test_results
-- Forcing gtest/gmock from source, though one was otherwise available.
-- Found gtest sources under '/usr/src/googletest': gtests will be built
-- Found gmock sources under '/usr/src/googletest': gmock will be built
-- Found PythonInterp: /usr/bin/python3 (found version "3.8.10")
-- Using Python nosetests: /usr/bin/nosetests3
-- catkin 0.8.10
-- BUILD_SHARED_LIBS is on
-- BUILD_SHARED_LIBS is on
-- ~~~~~ traversing 1 packages in topological order:
--   - hw0
-- ~~~~~
-- +++ processing catkin package: 'hw0'
-- => add_subdirectory(hw0)
-- Configuring done
-- Generating done
-- Build files have been written to: /home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/build
#####
##### Running command: "make -j6 -l6" in "/home/mahdi/Desktop/Robotics/project1/Hands_on/catkin_ws/build"
#####
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws$ 

```

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 فolder devel هست را source نکردم. پس مینویسیم :

. devel/setup.bash

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

```

mahdi@mahdi:~$ cd Desktop/Robotics/project1/Hands_on/catkin_ws/
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws$ ls
build  devel  src
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws$ rosrun hw0 sensor.py
[INFO] [1679090244.624450]: data: "hello guys"
[INFO] [1679090245.625697]: data: "hello guys"
[INFO] [1679090246.625639]: data: "hello guys"
[INFO] [1679090247.625688]: data: "hello guys"
[INFO] [1679090248.625753]: data: "hello guys"
[INFO] [1679090249.625629]: data: "hello guys"

```

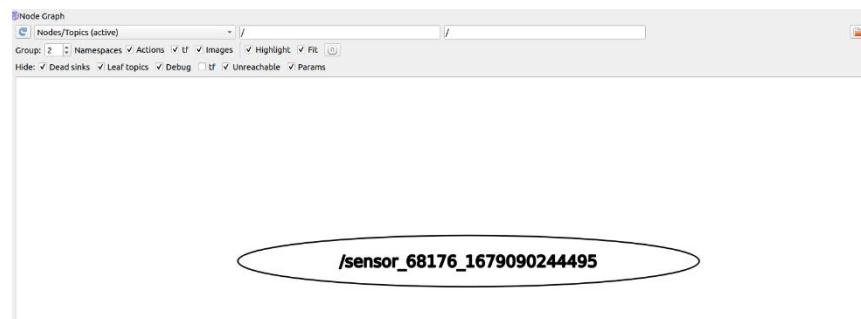
الان برای اینکه مطمئن شویم خود topic به درستی کار میکند یک rosnode list میزنیم و نود سنسور را میبینیم و همچنین در لیست تاپیکها، تاپیک هویج را نیز مشاهده خواهیم کرد:

```

mahdi@mahdi:~$ rosnode list
/rosout
/sensor_68176_1679090244495
mahdi@mahdi:~$ rostopic list
/havij
/rosout
/rosout_agg
mahdi@mahdi:~$ 

```

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

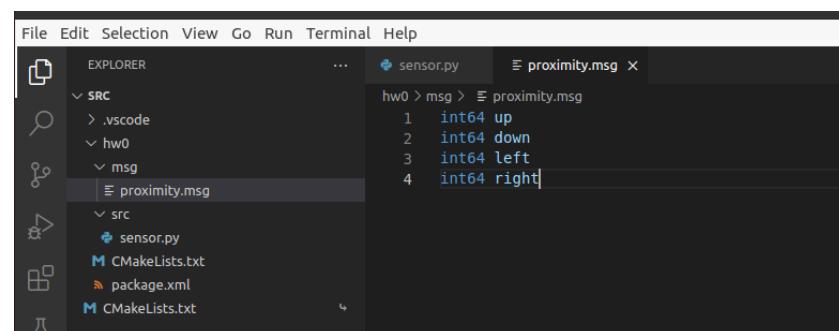


حال با توجه به صورت تمرین میخواهیم کمی کد را modify کنیم و تا جای ممکن نزدیک به نودی شود که برای تمرین صفر نیاز است.

اسم تاپیک را distance میگذاریم. نوع مسیج ۴ تا integer میباشد و string به درد ما نمیخورد. حالا رندوم عدد ایجاد کردن را نگفت ولی فعلاً میخواهیم custom message ایجاد کنیم. برای این کار وارد پوشه src میشویم و سپس وارد پکیج شده که در انجا یک فolder src داریم. یک فolder دیگه به اسم msg ایجاد میکنیم. همه فایل‌ای مسیج را قراره در این فolder اضافه کنیم.

```
mahdi@mahdi:~$ cd Desktop/Robotics/project1/Hands_on/catkin_ws/
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws$ ls
build  devel  src
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws$ cd src/
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src$ ls
CMakeLists.txt  hw0
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src$ cd hw0/
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src/hw0$ ls
CMakeLists.txt  package.xml  src
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src/hw0$ mkdir msg
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws/src/hw0$ █
```

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 ۴ تا مقدار عددی دارم و اعدادم هم integer اند.



برای اینکه این مسیح را توی فایل کد بشناسه، باید تغییراتی را داخل package.xml و CMakeLists.txt بدهیم. برای این منظور میتوانید به قسمت beginner tutorials در wiki.ros سطح مراجعه کنید و قسمت CreatingMsgAndSrv براساس آن قرار است پیش برویم. (لينك)

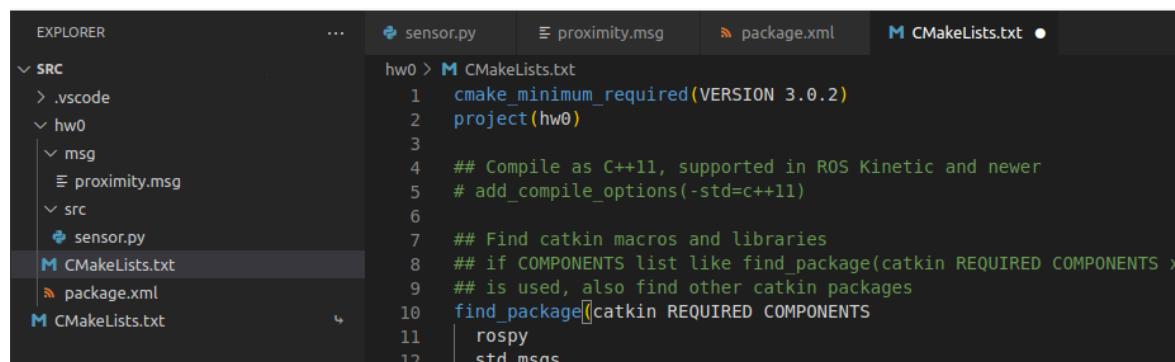
اولین تغییر در package.xml میباشد. دو خطی که در خطوط ۴۰ و ۴۶ هستند را uncomment کنید.

```

30      <!-- The *depend tags are used to specify dependencies -->
31      <!-- Dependencies can be catkin packages or system dependencies -->
32      <!-- Examples: -->
33      <!-- Use depend as a shortcut for packages that are both build and exec dependencies -->
34      <!--   <depend>roscpp</depend> -->
35      <!--   Note that this is equivalent to the following: -->
36      <!--     <build_depend>roscpp</build_depend> -->
37      <!--     <exec_depend>roscpp</exec_depend> -->
38      <!-- Use build_depend for packages you need at compile time: -->
39      <build_depend>message_generation</build_depend>
40      <!-- Use build_export depend for packages you need in order to build against this package: -->
41      <!--     <build_export_depend>message_generation</build_export_depend> -->
42      <!-- Use buildtool_depend for build tool packages: -->
43      <!--     <buildtool_depend>catkin</buildtool_depend> -->
44      <!-- Use exec_depend for packages you need at runtime: -->
45      <exec_depend>message_runtime</exec_depend>
46      <!-- Use test_depend for packages you need only for testing: -->
47      <!--     <test_depend>gtest</test_depend> -->
48      <!-- Use doc_depend for packages you need only for building documentation: -->
49      <!--     <doc_depend>doxygen</doc_depend> -->
50

```

حال باید به سراغ تغییرات داخل CMakeLists داخل hw0 هست) برویم. یکی اینکه در خط ۱۰ باید message_generation را اضافه نماییم.



The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows a tree view of the project structure under 'SRC'. It includes '.vscode', 'hw0' (which contains 'msg' and 'src' folders), and files like 'proximity.msg', 'sensor.py', 'package.xml', and two 'CMakeLists.txt' files (one in 'hw0' and one in the root).
- EDITOR:** Displays the content of the 'CMakeLists.txt' file in the 'hw0' directory. The code is as follows:

```

hw0 > M CMakeLists.txt
1 cmake_minimum_required(VERSION 3.0.2)
2 project(hw0)
3
4 ## Compile as C++11, supported in ROS Kinetic and newer
5 # add_compile_options(-std=c++11)
6
7 ## Find catkin macros and libraries
8 ## if COMPONENTS list like find_package(catkin REQUIRED COMPONENTS x
9 ## is used, also find other catkin packages
10 find_package(catkin REQUIRED COMPONENTS
11   rospy
12   std_msgs

```

تغییر دیگر در خط ۱۰۵ catkin_package میباشد:

```

96 #####
97 ## catkin specific configuration ##
98 #####
99 ## The catkin_package macro generates cmake config files for your package
100 ## Declare things to be passed to dependent projects
101 ## INCLUDE_DIRS: uncomment this if your package contains header files
102 ## LIBRARIES: libraries you create in this project that dependent projects also need
103 ## CATKIN_DEPENDS: catkin_packages dependent projects also need
104 ## DEPENDS: system dependencies of this project that dependent projects also need
105 catkin_package(
106   CATKIN_DEPENDS message_runtime
107 )

```

سپس به دنبال add_message_files میگردیم و آن را uncomment میکنیم و جای Message2.msg و Message1.msg میگذاریم.

```

49     ## Generate messages in the 'msg' folder
50     add_message_files(
51         FILES
52         proximity.msg
53     )
54

```

حال باید comment را از generate_message در بیاریم.

```

69     ## Generate added messages and services with any dependencies listed here
70     generate_messages([
71         DEPENDENCIES
72         std_msgs
73     ])
74

```

کار دیگری که باید بکنیم اینه که در خط ۱۱۵ اون include را هم uncomment کنیم.

```

112     ## Specify additional locations of header files
113     ## Your package locations should be listed before other locations
114     include_directories(
115         include
116         ${catkin_INCLUDE_DIRS}
117     )
118

```

چون دوباره پکیج جدید درست کردیم باید catkin_make را صدا بکنیم.(در همان ترمینالی که فعال بود ابتدا با Ctrl+C آن را از کار 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 متدهای publisher هم تایپ را proximity میگذاریم. در گام بعد باید message داخل تاپیک میگذاریم را داخل لوپ عوض کنیم. یک مسیج از جنس proximity درست میکنیم و بعد مادری متغیرهای up، down و ... را ست میکنیم.

```

sensor.py x  proxy.msg  package.xml  CMakeLists.txt
hw0 > src > sensor.py > talker
1   #!/usr/bin/python3
2
3   import rospy
4   from std_msgs.msg import String
5   from hw0.msg import proximity
6
7   def talker():
8       pub = rospy.Publisher("distance", proximity, queue_size= 10)
9       rospy.init_node("sensor", anonymous=True)
10      rate = rospy.Rate(1) #Hz
11
12      while not rospy.is_shutdown():
13          msg = proximity()
14          msg.up = 10
15          msg.down = 20
16          msg.left = 30
17          msg.right = 40
18
19          rospy.loginfo(msg)
20          pub.publish(msg)
21          rate.sleep()
22
23  if __name__ == "__main__":
24      talker()

```

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 rosmg show proximity را میزنیم که میبینیم hw0/ وجود دارد. سپس با نمایش مسیج تاپیک show خواهیم دید:

```
mahdi@mahdi:~$ cd Desktop/Robotics/project1/Hands_on/catkin_ws/  
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws$ . devel/setup.bash  
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws$ rosmg show  
^Cmahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws$ rosmg show hw0/proximity  
int64 up  
int64 down  
int64 left  
int64 right  
  
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws$
```

حال میتوان نود موردنظر را اجرا کرد:

```
mahdi@mahdi:~/Desktop/Robotics/project1/Hands_on/catkin_ws$ rosrun hw0 sensor.py  
[INFO] [1679095416.047246]: up: 10  
down: 20  
left: 30  
right: 40  
[INFO] [1679095417.048798]: up: 10  
down: 20  
left: 30  
right: 40  
[INFO] [1679095418.047307]: up: 10  
down: 20  
left: 30  
right: 40  
[INFO] [1679095419.048476]: up: 10  
down: 20  
left: 30  
right: 40  
[INFO] [1679095420.048737]: up: 10  
down: 20
```

بخش ۳ - مراحل ساختن نودهای خواسته شده در دستور کار

• ساختن نود سنسور

در ابتدای کار لازم است تا یک work space بسازیم. من یک work space با نام hw1_ws میسازم. سپس به فolder src در آن رفته و workspace را initialize میکنم.

```
mahdi@mahdi:~/Desktop/Robotics/project1/$ cd Desktop/Robotics/project1/
mahdi@mahdi:~/Desktop/Robotics/project1$ mkdir -p hw1_ws/src
mahdi@mahdi:~/Desktop/Robotics/project1$ cd hw1_ws/
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ ls
src
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ cd src/
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src$ ls
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src$ catkin_init_workspace
Creating symlink "/home/mahdi/Desktop/Robotics/project1/hw1_ws/src/CMakeLists.txt"
pointing to "/opt/ros/noetic/share/catkin/cmake/toplevel.cmake"
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src$ ls
CMakeLists.txt
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src$ cd ..
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ █
```

حال باید دستور catkin_make را اجرا کنیم. و بعد اگر ls را بزنیم خواهیم دید دو فolder build و devel اضافه شدند.

```
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ catkin_make
Base path: /home/mahdi/Desktop/Robotics/project1/hw1_ws
Source space: /home/mahdi/Desktop/Robotics/project1/hw1_ws/src
Build space: /home/mahdi/Desktop/Robotics/project1/hw1_ws/build
Devel space: /home/mahdi/Desktop/Robotics/project1/hw1_ws/devel
Install space: /home/mahdi/Desktop/Robotics/project1/hw1_ws/install
#####
#### Running command: "cmake /home/mahdi/Desktop/Robotics/project1/hw1_ws/src -DCATKIN_DEVEL_PREFIX=/home/mahdi/Desktop/Robotics/project1/hw1_ws/install -G Unix Makefiles" in "/home/mahdi/Desktop/Robotics/project1/hw1_ws/build"
#####
-- The C compiler identification is GNU 9.4.0
-- The CXX compiler identification is GNU 9.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ ls
build  devel  src
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ █
```

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 add dependency را میکنیم مثلًا میدانیم در این پکیج باید از rospy و std_msgs استفاده کنیم. داخل این پکیج ساخته شده میرویم و اگر ls بزنیم ۳ تا اسم نمایش میدهد.

از میان آن‌ها ما کدهامون را داخل فolder src قرار میدهیم. برای نوشتن کد داخل فolder 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.

```
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src$ catkin_create_pkg my_turtle rospy std_msgs
Created file my_turtle/package.xml
Created file my_turtle/CMakeLists.txt
Created folder my_turtle/src
Successfully created files in /home/mahdi/Desktop/Robotics/project1/hw1_ws/src/my_turtle. Please adjust the values in package.xml.
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src$ ls
CMakeLists.txt my_turtle
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src$ cd my_turtle/
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle$ ls
CMakeLists.txt package.xml src
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle$ cd src/
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle/src$ touch sensor.py
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle/src$ ls
sensor.py
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle/src$ chmod +x sensor.py
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle/src$ ls
sensor.py
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle/src$
```

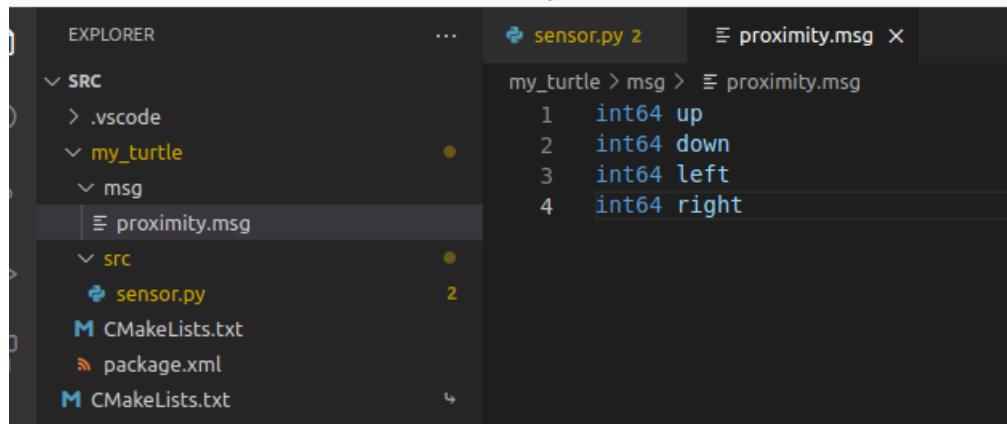
مراحل بعدی را از داخل vscode انجام میدهیم. در ترمینال بک میزنیم تا به پوشه src اول برسیم.

```
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle/src$ cd ..
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle$ cd ..
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src$ code .
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src$
```

برای کد زدن لازم است که از message type مورد نظر استفاده کنیم. پس بهتر است برویم و در همین ابتدای کار custom message را بسازیم. برای این کار وارد پوشه src اول میشویم و سپس وارد پکیج شده که در انجا یک فolder src داریم. یک فolder دیگه به اسم msg ایجاد میکنیم. همه فایل‌ی مسیج را قراره در این فolder اضافه کنیم.

```
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src$ ls
CMakeLists.txt my_turtle
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src$ cd my_turtle/
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle$ ls
CMakeLists.txt package.xml src
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle$ mkdir msg
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle$ ls
CMakeLists.txt msg package.xml src
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle$
```

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 ۴ 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 beginner tutorials در wiki.ros مراجعه میکنیم و قسمت CreatingMsgAndSrv میرویم. ([لينك](#)) براساس آن قرار است پیش برویم.

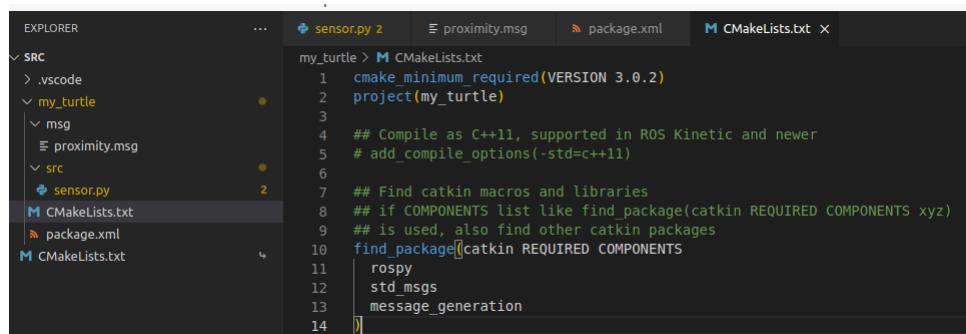
اولین تغییر در package.xml میباشد. دو خطی که در خطوط ۴۰ و ۴۶ هستند را uncomment کنید.

```

30      <!-- The *depend tags are used to specify dependencies -->
31      <!-- Dependencies can be catkin packages or system dependencies -->
32      <!-- Examples: -->
33      <!-- Use depend as a shortcut for packages that are both build and exec dependencies -->
34      <!--    <depend>roscpp</depend> -->
35      <!--    Note that this is equivalent to the following: -->
36      <!--    <build_depend>roscpp</build_depend> -->
37      <!--    <exec_depend>roscpp</exec_depend> -->
38      <!-- Use build_depend for packages you need at compile time: -->
39      <build_depend>message_generation</build_depend>
40      <!-- Use build_export_depend for packages you need in order to build against this package: -->
41      <!--    <build_export_depend>message_generation</build_export_depend> -->
42      <!-- Use buildtool_depend for build tool packages: -->
43      <!--    <buildtool_depend>catkin</buildtool_depend> -->
44      <!-- Use exec_depend for packages you need at runtime: -->
45      <exec_depend>message_runtime</exec_depend>
46      <!-- Use test_depend for packages you need only for testing: -->
47      <!--    <test_depend>gtest</test_depend> -->
48      <!-- Use doc_depend for packages you need only for building documentation: -->
49      <!--    <doc_depend>doxygen</doc_depend> -->
50

```

حال باید به سراغ تغییرات داخل CMakeLists.txt (اویی که داخل پکیج my_turtle هست) برویم. یکی اینکه در خط ۱۰ باید message_generation را اضافه نماییم.



تغییر دیگر در خط ۱۰۵ catkin_package میباشد:

```
96 #####  
97 ## catkin specific configuration ##  
98 #####  
99 ## The catkin_package macro generates cmake config files for your package  
100 ## Declare things to be passed to dependent projects  
101 ## INCLUDE_DIRS: uncomment this if your package contains header files  
102 ## LIBRARIES: libraries you create in this project that dependent projects also need  
103 ## CATKIN_DEPENDS: catkin_packages dependent projects also need  
104 ## DEPENDS: system dependencies of this project that dependent projects also need  
105 catkin_package(  
106 | CATKIN_DEPENDS message_runtime  
107 )  
108
```

سپس به دنبال proximity.msg و آن را uncomment میکنیم و جای Message2.msg و Message1.msg میگذاریم.

```
49 ## Generate messages in the 'msg' folder  
50 add_message_files(  
51 | FILES  
52 | proximity.msg  
53 )  
54
```

حال باید comment را از generate_message در بیاریم.

```
69 ## Generate added messages and services with any dependencies listed here  
70 generate_messages(  
71 | DEPENDENCIES  
72 | std_msgs  
73 )  
74
```

کار دیگری که باید بکنیم اینه که در خط ۱۱۵ اون include را هم uncomment کنیم.

```
112 ## Specify additional locations of header files  
113 ## Your package locations should be listed before other locations  
114 include_directories(  
115 | include  
116 | ${catkin_INCLUDE_DIRS}  
117 )  
118
```

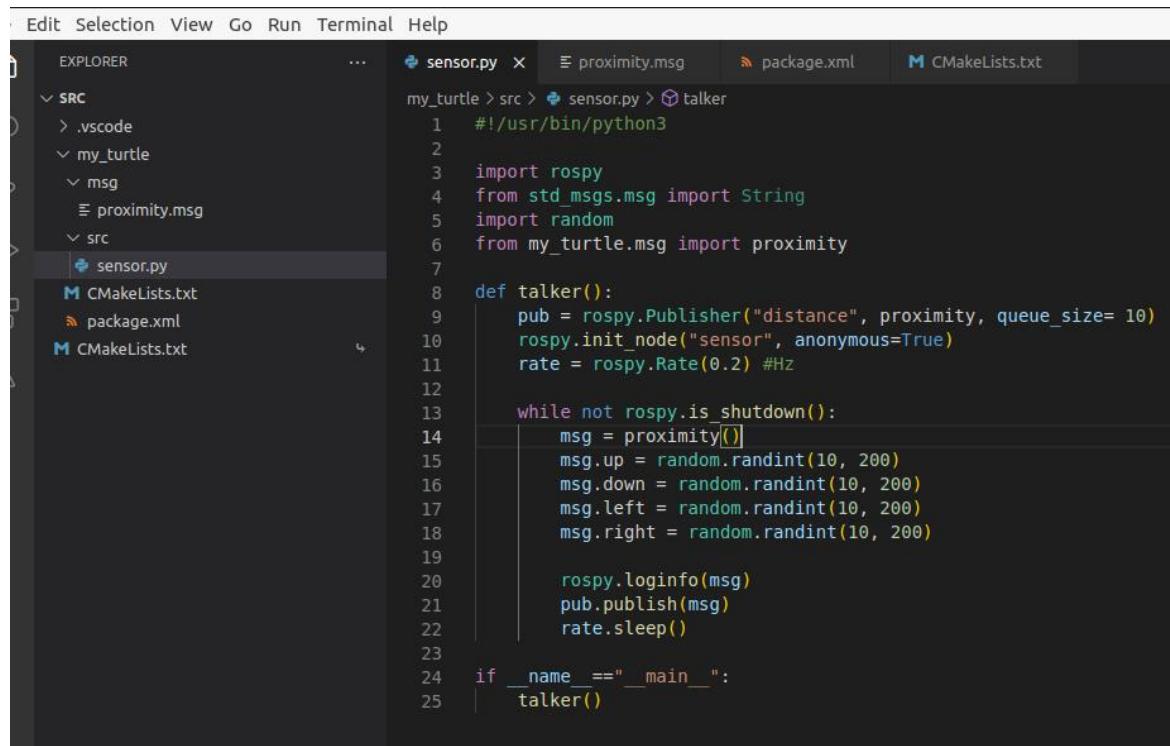
چون دوباره پکیج جدید درست کردیم باید catkin_make را صدا بکنیم.

```
mhadi@mhadi:~/Desktop/Robotics/project1/hw1_ws$ catkin_make
Base path: /home/mahdi/Desktop/Robotics/project1/hw1_ws
Source space: /home/mahdi/Desktop/Robotics/project1/hw1_ws/src
Build space: /home/mahdi/Desktop/Robotics/project1/hw1_ws/build
Devel space: /home/mahdi/Desktop/Robotics/project1/hw1_ws/devel
Install space: /home/mahdi/Desktop/Robotics/project1/hw1_ws/install
#####
##### Running command: "make cmake_check_build_system" in "/home/mahdi/Desktop/Robotics/project1/hw1_ws/build"
#####
-- Using CMAKE_PREFIX_PATH: /home/mahdi/Desktop/Robotics/project1/hw1_ws/devel
-- Using CMAKE_PREFIX_PATH: /opt/ros/noetic
-- This workspace overlays: /opt/ros/noetic
-- Found PythonInterp: /usr/bin/python3 (found suitable version "3.8.10", minimum required is "3")
-- Using PYTHON_EXECUTABLE: /usr/bin/python3
-- Using Debian Python package layout
-- Using empy: /usr/lib/python3/dist-packages/empy
-- Using CATKIN_ENABLE_TESTING: ON
-- Call enable_testing()
-- Using CATKIN_TEST_RESULTS_DIR: /home/mahdi/Desktop/Robotics/project1/hw1_ws/build/test_results
-- Forcing gtest/gmock from source, though one was otherwise available.
-- Found gtest sources under '/usr/src/googletest': gtests will be built
-- Found gmock sources under '/usr/src/googletest': gmock will be built
-- Found PythonInterp: /usr/bin/python3 (found version "3.8.10")
-- Found PythonNoseTester: /usr/bin/nosetests3
-- Using PythonNoseTester: /usr/bin/nosetests3
catkin 0.8.10
-- BUILD_SHARED_LIBS is on
-- BUILD_SHARED_LIBS is on
...
--   traversing 1 packages in topological order:
--   - my_turtle
...
--   +++ processing catkin package: 'my_turtle'
--   =>>> add_subdirectory(my_turtle)
--   Using these message generators: gencpp;geneus;genlisp;genodejs;genpy
-- my_turtle: 1 messages, 0 services
-- Configuring done
-- Generating done
-- Build files have been written to: /home/mahdi/Desktop/Robotics/project1/hw1_ws/build
#####
##### Running command: "make -j6 -l6" in "/home/mahdi/Desktop/Robotics/project1/hw1_ws/build"
#####
Scanning dependencies of target _my_turtle_generate_messages_check_deps_proximity
Scanning dependencies of target std_msgs_generate_messages_cpp
[  0%] [  0%]
```

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

from my_turtle.msg import proximity

کد موردنظر به صورت زیر میشود:



The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure:
 - SRC
 - .vscode
 - my_turtle
 - msg
 - proximity.msg
 - src
 - sensor.py
 - CMakeLists.txt
 - package.xml
 - CMakeLists.txt- Code Editor:** Displays the content of `sensor.py`:

```
#!/usr/bin/python3
import rospy
from std_msgs.msg import String
import random
from my_turtle.msg import proximity

def talker():
    pub = rospy.Publisher("distance", proximity, queue_size= 10)
    rospy.init_node("sensor", anonymous=True)
    rate = rospy.Rate(0.2) #Hz

    while not rospy.is_shutdown():
        msg = proximity()
        msg.up = random.randint(10, 200)
        msg.down = random.randint(10, 200)
        msg.left = random.randint(10, 200)
        msg.right = random.randint(10, 200)

        rospy.loginfo(msg)
        pub.publish(msg)
        rate.sleep()

if __name__ == "__main__":
    talker()
```
- Terminal:** The terminal tab is visible at the top right of the interface.

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 show و سپس اسم تاپیک را مینویسیم و سپس خواهیم نوشت:

```
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ cd Desktop/Robotics/project1/hw1_ws/
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ roscore
... logging to /home/mahdi/.ros/log/4645e0b4-c584-11ed-a4b0-b7c675675d8f/roslaun
ch-mahdi-6982.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://mahdi:46473/
ros_comm version 1.15.15

SUMMARY
=====
PARAMETERS
  * /rostdistro: noetic
  * /rosversion: 1.15.15

NODES
auto-starting new master
process[master]: started with pid [6992]
ROS_MASTER_URI=http://mahdi:11311/

setting /run_id to 4645e0b4-c584-11ed-a4b0-b7c675675d8f
process[rosout-1]: started with pid [7002]
  started core service [/rosout]

mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ rosmsg show my_turtle/proximity
int64 up
int64 down
int64 left
int64 right

mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ 
```

حال میتوان نود موردنظر را اجرا کرد:

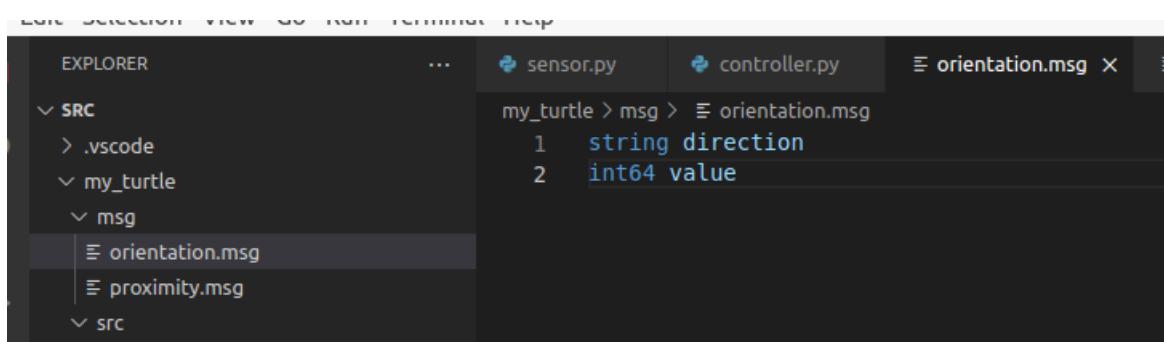
```
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ rosrun my_turtle sensor.py
[INFO] [1679141061.485401]: up: 181
down: 46
left: 83
right: 81
[INFO] [1679141066.487459]: up: 21
down: 35
left: 90
right: 192
[INFO] [1679141071.490697]: up: 100
down: 70
left: 120
right: 125
```

• ساختن نود کنترلر

برای این منظور در داخل فolder src در پکیج my_turtle لازم است تا controller.py را ایجاد کنیم و همچنین آن را به کمک chmod تبدیل به یک فایل executable میکنیم.

```
^Cmahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ ls
build  devel  src
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ cd src/
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src$ ls
CMakeLists.txt  my_turtle
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src$ cd my_turtle/
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle$ cd src/
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle/src$ touch controller.py
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle/src$ ls
controller.py  sensor.py
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle/src$ chmod +x controller.py
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle/src$ ls
controller.py  sensor.py
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle/src$ cd ..
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle$ cd ..
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src$ code .
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src$ █
```

حال بقیه موارد را از داخل vscode هندل میکنیم. ابتدا قبل از اینکه کد زنی را شروع کنیم مربوطه را میسازیم. در vscode روی msg کلیک راست کرده و یک new file ایجاد کرده و اسم آن را orientation.msg میگذاریم و سپس در آن لازم است دو متغیر تعریف کنیم. یک متغیر direction از جنس string میباشد و این متغیر میتواند ۳ تا مقدار clockwise و counter clockwise باشد. دو دیگر int64 میباشد که از جنس value میباشد. متغیر dont rotate بگیرد. متغیر direction میباشد که از جنس int64 است و درواقع مقدار آن ۰ یا ۹۰ یا ۲۷۰ میباشد که میزان چرخش به درجه است.



حال باید در فایل CMakeLists.txt مربوط به پکیج my_turtle را اصلاح زیر را انجام دهیم.

```
49  ## Generate messages in the 'msg' folder
50  add_message_files(
51  |   FILES
52  |   proximity.msg
53  |   orientation.msg
54  )
```

در گام بعد لازم است که catkin_make را بزنیم.

```
[12:11:17 09/09/2021] [12:11:17 09/09/2021] [12:11:17 09/09/2021]
^Cmahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ catkin_make
Base path: /home/mahdi/Desktop/Robotics/project1/hw1_ws
Source space: /home/mahdi/Desktop/Robotics/project1/hw1_ws/src
Build space: /home/mahdi/Desktop/Robotics/project1/hw1_ws/build
Devel space: /home/mahdi/Desktop/Robotics/project1/hw1_ws/devel
Install space: /home/mahdi/Desktop/Robotics/project1/hw1_ws/install
#####
##### Running command: "make cmake_check_build_system" in "/home/mahdi/Desktop/Robotics/project1/hw1_ws/build"
#####
-- Using CATKIN_DEVEL_PREFIX: /home/mahdi/Desktop/Robotics/project1/hw1_ws/devel
-- Using CMAKE_PREFIX_PATH: /home/mahdi/Desktop/Robotics/project1/hw1_ws/devel;/opt/ros/noetic
-- This workspace overlays: /home/mahdi/Desktop/Robotics/project1/hw1_ws/devel;/opt/ros/noetic
-- Found PythonInterp: /usr/bin/python3 (found suitable version "3.8.10", minimum required is "3")
-- Using PYTHON_EXECUTABLE: /usr/bin/python3
-- Using Deblian Python package layout
-- Using empy: /usr/lib/python3/dist-packages/emp.py
-- Using CATKIN_ENABLE_TESTING: ON
-- Call enable_testing()
-- Using CATKIN_TEST_RESULTS_DIR: /home/mahdi/Desktop/Robotics/project1/hw1_ws/build/test_results
-- Forcing gtest/gmock from source, though one was otherwise available.
-- Found gtest sources under '/usr/src/googletest': gtests will be built
-- Found gmock sources under '/usr/src/googletest': gmock will be built
-- Found PythonInterp: /usr/bin/python3 (found version "3.8.10")
-- Using Python nosetests: /usr/bin/nosetests3
-- catkin 0.8.10
-- BUILD_SHARED_LIBS is on
-- BUILD_SHARED_LIBS is on
-- ~~~ traversing 1 packages in topological order:
-- ~~~ - my_turtle
-- ~~~
-- +++ processing catkin package: 'my_turtle'
-- ==> add_subdirectory(my_turtle)
-- Using these message generators: genCPP;geneus;genlisp;gennodejs;genpy
-- my_turtle: 2 messages, 0 services
-- Configuring done
-- Generating done
-- Build files have been written to: /home/mahdi/Desktop/Robotics/project1/hw1_ws/build
#####
##### Running command: "make -j6 -l6" in "/home/mahdi/Desktop/Robotics/project1/hw1_ws/build"
#####
Scanning dependencies of target _my_turtle_generate_messages_check_deps_orientation
[ 0%] Built target std_msgs_generate_messages_cxx
```

حال به سراغ کد مربوط به فایل controller.py میرویم. در اینجا publisher و subscriber میباشد. ابتدا یک فانکشن با نام () initialize() تعریف میکنیم و در آن نود را میکنیم و اسم نود را هم controller میگذاریم. سپس دو متغیر گلوبال pub1 و pub2 تعریف میکنیم. سپس initialization را انجام میدهیم. دقت شود pub1 قرار هست روی تاپیک motor1 را publish کند و pub2 روی تاپیک motor2 پابلیش کند. سپس subscriber را initialize میکنیم و میگوییم که تاپیک distance را subscribe کند. یک متده است که عنوان callback پاس میدهیم که هر بار پیامی روی تاپیک مربوطه پابلیش شود، آن هم اجرا میشود. (Rospy.spin() هم باعث میشود که تا زمانی که node متوقف نشده این کد اجرا شود).

حال به سراغ متده callback میرویم. ابتدا لازم است دو متغیر گلوبال state که نشاندهنده جهت قرار گیری ربات (مثلابه سمت شمال میرود یا غرب یا...) است و متغیر states که یک لیست از وضعیت‌های ممکن برای متغیر state میباشد. در این متده callback ابتدا به کمک loginfo استیت اولیه ربات و سپس پیامی که از تاپیک دریافت کرده است را نشان میدهد. سپس لازم است که message ای که قرار است پس از آنالیز مقدار دیتای ورودی، بسازیم و publish کنیم را تعریف و مقداردهی اولیه کنیم. به صورت دیفالت

وقتی لازم نباشد چرخشی انجام دهیم و پشت ربات به جای درستی باشد، ما باید مقدارهای dont rotate و ۰ را به ترتیب برای direction و value پابلیش کنیم. لازم به ذکر است که ما فرض کردیم down همان پشت ربات میباشد. سپس یک متغیر distances شامل مقادیر خوانده شده از تاپیک تعريف میشود به نحوی که مقادیر ساعتگرد چیده شوند و ایندکس آنها به گونه‌ای باشد که اگر بخواهیم ایندکس حالت down (یا همان back) از آنها کم کنیم علامت و مقدار این تفریق بیانگر تعداد ۹۰ درجه ها و جهت چرخش باشد. چنانچه مینیمم فاصله در جهتی غیر از down باشد باید بچرخیم و زاویه را حساب کنیم و مقادیر direction و value را آپدیت کنیم.

دقت شود فرض کردیم مقدار اولیه state north ربات میباشد یعنی درحال حرکت به سمت شمال میباشد. حال بعد از آپدیت کردن مقدار فیلدهای message باید مقدار state را هم آپدیت کنیم.

در نهایت msg که درست کردیم در هردو تاپیک یعنی تاپیک motor1 و motor2 پابلیش میکنیم. در ادامه اسکرین شات کد قرار گرفته است.

```

sensor.py controller.py x motor1.py motor2.py orientation.msg proximity.msg package.xml M
my_turtle > src > controller.py > ↗ callback
1  #!/usr/bin/python3
2
3 import rospy
4 from std_msgs.msg import String
5 from my_turtle.msg import proximity
6 from my_turtle.msg import orientation
7
8 # some global variables
9 state = "north"
10 states = ["east", "south", "west", "north"]
11
12 pub1 = None
13 pub2 = None
14
15 def callback(data):
16     global state, states
17     rospy.loginfo("\nprevious state: "+state)
18     rospy.loginfo("received data in controller:\n%s", data)
19
20     # we can define message and initialize it with some default value
21     # if the obstacle nearest to the down(back) side these deafault values remain without any changes
22     msg = orientation()
23     msg.direction = "dont rotate"
24     msg.value = 0
25
26     # I consider that down means back of robot
27     # sequence in below list is important and they are placed clockwise
28     # and also I want differences in their index show us the direction
29     distances = [data.right, data.down, data.left, data.up]
30     min_distance = min(distances)
31     min_distance_index = distances.index(min_distance)
32
33     if min_distance != distances[1]:
34         # direction coef tell us howmany 90 degrees in which direction we should rotate
35         direction_coefficient = min_distance_index - 1

```

```

37     # assign direction to msg.direction variable
38     if direction_coefficient < 0:
39         msg.direction = "counter clockwise"
40     else:
41         msg.direction = "clockwise"
42
43     # assign value of orientation in degree to msg.value
44     msg.value = abs(direction_coefficient) * 90
45
46     # update state of robot after orientation
47     state = states[(states.index(state)+direction_coefficient)%4]
48
49     rospy.loginfo("new state:"+state)
50
51     # now we should publish message to both motors
52     pub1.publish(msg)
53     pub2.publish(msg)
54
55 def listener_talker():
56     global pub1, pub2
57     # In ROS, nodes are uniquely named. If two nodes with the same
58     # name are launched, the previous one is kicked off. The
59     # anonymous=True flag means that rospy will choose a unique
60     # name for our 'controller' node so that multiple controllers can
61     # run simultaneously.
62     rospy.init_node('controller', anonymous=True)
63
64     pub1 = rospy.Publisher("motor1", orientation, queue_size= 10)
65     pub2 = rospy.Publisher("motor2", orientation, queue_size= 10)
66
67     rospy.Subscriber("distance", proximity, callback)
68
69     # spin() simply keeps python from exiting until this node is stopped
70     rospy.spin()
71
72 if __name__ == '__main__':
73     listener_talker()

```

حال میتوانیم تست کنیم که ببینیم آیا فرآیند subscribe کردن و تصمیم گیری به درستی انجام میشود یا خیر. یک ترمینال باز کرده و در آن roscore را میزنیم. یک ترمینال دیگر برای نod سنسور و یک ترمینال هم برای نod کنترلر باز میکنیم. همانطور که در زیر مشاهده میشود دیتاهای درستی دریافت شده و همچنین زمانی که پشت ربات به درستی به سمت کمترین فاصله است تغییری در state نداریم و در حالتی که نیاز به چرخش است به درستی این چرخش انجام میشود.

```

mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ cd Desktop/Robotics/project1/hw1_ws/
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ . devel/setup.bash
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ rossrun my_turtle sensor.py
[INFO] [1679161033.295261]: published data from sensor:
up: 194
down: 123
left: 151
right: 189

[INFO] [1679161038.300541]: published data from sensor:
up: 28
down: 173
left: 56
right: 105

[INFO] [1679161043.297333]: published data from sensor:
up: 22
down: 75
left: 40
right: 154

mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ cd Desktop/Robotics/project1/hw1_ws/
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ . devel/setup.bash
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ rossrun my_turtle controller.py
[INFO] [1679161033.296352]: previous state: north
[INFO] [1679161033.297479]: received data in controller:
up: 194
down: 123
left: 151
right: 189
[INFO] [1679161033.298598]: new state: north

[INFO] [1679161038.303052]: previous state: north
[INFO] [1679161038.304021]: received data in controller:
up: 28
down: 173
left: 56
right: 105
[INFO] [1679161038.304828]: new state: south

[INFO] [1679161043.298826]: previous state: south
[INFO] [1679161043.300344]: received data in controller:
up: 22
down: 75
left: 40
right: 154
[INFO] [1679161043.301808]: new state: north

```

• ساختن نودهای motor2 و motor1

برای این منظور در داخل فolder src در پکیج my_turtle لازم است تا motor1.py را ایجاد کنیم و همچنین آن را به کمک chmod تبدیل به یک فایل executable میکنیم. همین کار را برای motor2.py انجام میدهیم.

```
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ cd src/
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src$ cd my_turtle/
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle$ cd src/
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle/src$ touch motor1.py
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle/src$ chmod +x motor1.py
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle/src$ touch motor2.py
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle/src$ chmod +x motor2.py
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle/src$ ls
controller.py  motor1.py  motor2.py  sensor.py
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws/src/my_turtle/src$
```

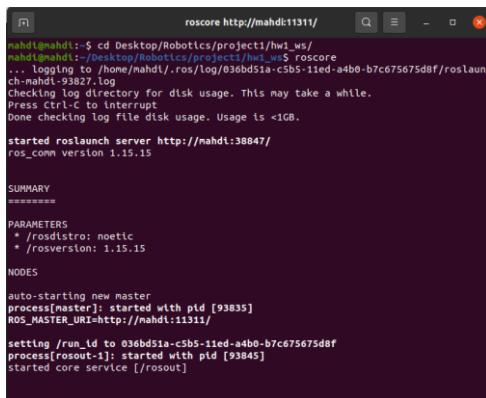
در این نودها هم motor1 و هم motor2 هردو فقط subscriber هستند. motor1 باید تاپیک motor2 را کند و در motor2 باید تاپیک motor1 subscribe کند. جنس message‌هایی که دریافت میکنند orientation میباشد. کد این دو نود بسیار شبیه به هم میباشد و به صورت زیر هستند:

```
sensor.py controller.py motor1.py ✘ motor2.py orientation.msg
my_turtle > src > motor1.py > ...
1 #!/usr/bin/python3
2
3 import rospy
4 from my_turtle.msg import orientation
5
6 def callback(data):
7     rospy.loginfo("received data in motor1\n %s", data)
8
9 def listener():
10    # In ROS, nodes are uniquely named. If two nodes with the same
11    # name are launched, the previous one is kicked off. The
12    # anonymous=True flag means that rospy will choose a unique
13    # name for our 'motor1' node so that multiple motor1s can
14    # run simultaneously.
15    rospy.init_node('motor1', anonymous=True)
16
17    rospy.Subscriber("motor1", orientation, callback)
18
19    # spin() simply keeps python from exiting until this node is stopped
20    rospy.spin()
21
22 if __name__ == '__main__':
23     listener()

my_turtle > src > motor2.py > ...
1 #!/usr/bin/python3
2
3 import rospy
4 from my_turtle.msg import orientation
5
6 def callback(data):
7     rospy.loginfo("received data in motor2\n %s", data)
8
9 def listener():
10    # In ROS, nodes are uniquely named. If two nodes with the same
11    # name are launched, the previous one is kicked off. The
12    # anonymous=True flag means that rospy will choose a unique
13    # name for our 'motor2' node so that multiple motor2s can
14    # run simultaneously.
15    rospy.init_node('motor2', anonymous=True)
16
17    rospy.Subscriber("motor2", orientation, callback)
18
19    # spin() simply keeps python from exiting until this node is stopped
20    rospy.spin()
21
22 if __name__ == '__main__':
23     listener()
```

بخش ۴ - صحت سنجی نودها و اسکرین شات از آن‌ها

حال لازم است تا همه‌ی نودها را اجرا کنیم و درستی روای انجام شده را بررسی کنیم. برای این منظور لازم است تا یک ترمینال باز کرده و master را با دستور roscore در آن اجرا کنیم.



```
roscore http://mahdi:11311/
mahdi@mahdi: ~ cd Desktop/Robotics/project1/hw1_ws/
mahdi@mahdi: ~/Desktop/Robotics/project1/hw1_ws$ roscore
...
Logging to /home/mahdi/.ros/log/036bd51a-c5b5-11ed-a4b0-b7c675675d8f/roslaun
ch-mahdi-93827.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://mahdi:38847/
ros_comm version 1.15.15

SUMMARY
=====
PARAMETERS
  * /rosdistro: noetic
  * /rosversion: 1.15.15

NODES
auto-starting new master
process[master]: started with pid [93835]
ROS_MASTER_URI=http://mahdi:11311/
setting /run_id to 036bd51a-c5b5-11ed-a4b0-b7c675675d8f
process[rosout-1]: started with pid [93845]
started core service [/rosout]
```

حال ۴ ترمینال برای نودهای مختلف باز میکنیم و کامندهای زیر را به ترتیب اجرا میکنیم:

برای موتور ۱ :

- cd Desktop/Robotics/project1/hw1_ws/
- . devel/setup.bash
- rosrun my_turtle motor1.py

برای موتور ۲ :

- cd Desktop/Robotics/project1/hw1_ws/
- . devel/setup.bash
- rosrun my_turtle motor2.py

برای کنترلر :

- cd Desktop/Robotics/project1/hw1_ws/
- . devel/setup.bash
- rosrun my_turtle controller.py

برای سنسور :

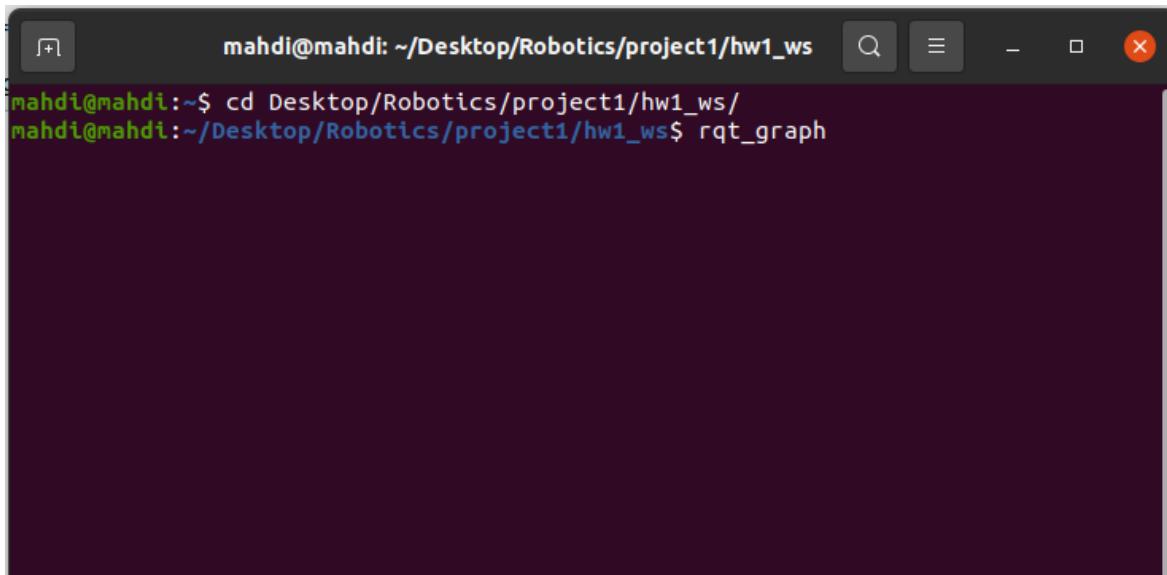
- cd Desktop/Robotics/project1/hw1_ws/
- . devel/setup.bash
- rosrun my_turtle sensor.py

برای مثال چندین iteration را در زیر میتوانید ببینید که در آن همه حالت ممکن قابل رویت است:

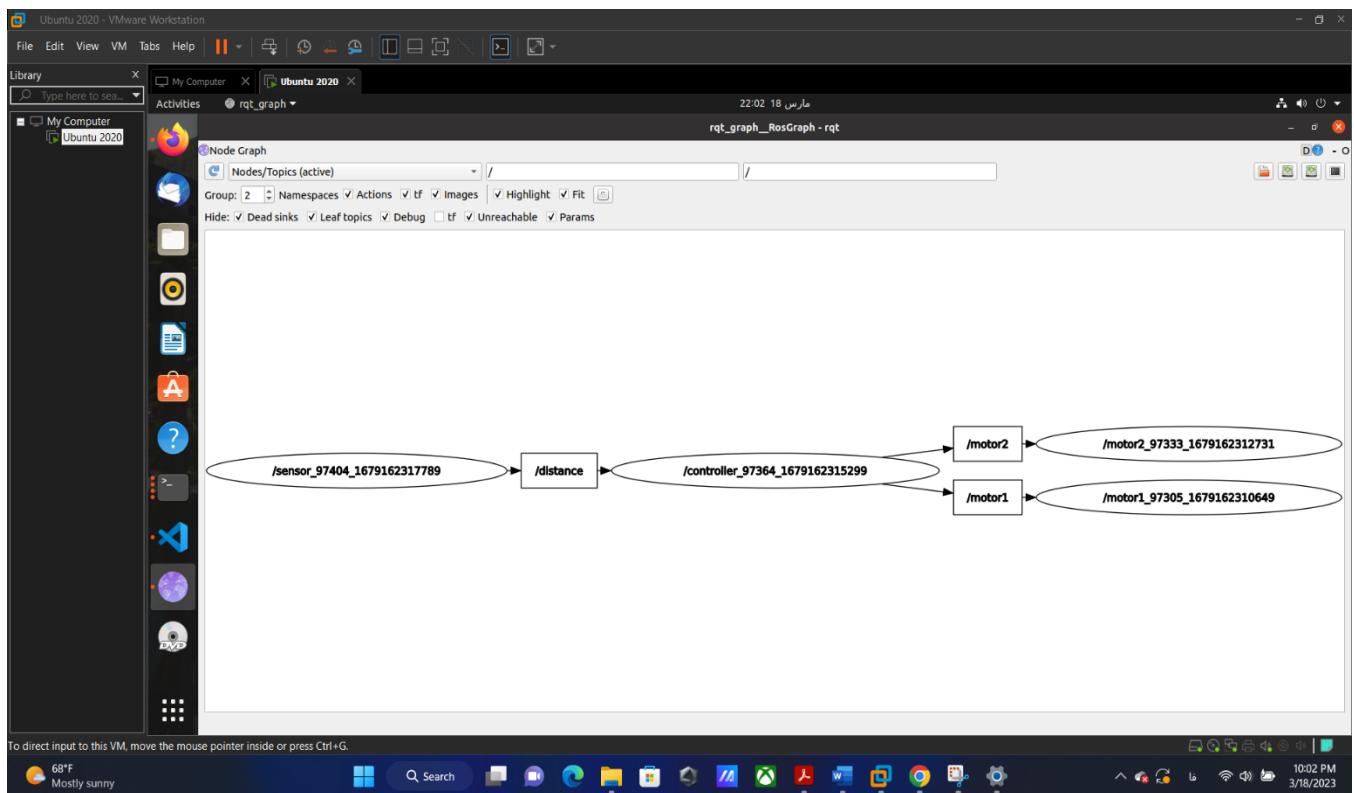
```
mahdi@mahdi: ~/Desktop/Robotics/project1/hw1_ws
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ cd Desktop/Robotics/project1/hw1_ws/
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ . devel/setup.bash
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ rosrun my_turtle sensor.py
[INFO] [1679162317.917058]: published data from sensor:
up: 26
down: 123
left: 89
right: 72
[INFO] [1679162322.922439]: published data from sensor:
up: 183
down: 58
left: 185
right: 115
[INFO] [1679162327.922416]: published data from sensor:
up: 78
down: 123
left: 92
right: 139
mahdi@mahdi: ~/Desktop/Robotics/project1/hw1_ws
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ . devel/setup.bash
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ rosrun my_turtle controller.py
[INFO] [1679162317.917058]: previous state: north
[INFO] [1679162317.919025]: received data in controller:
up: 26
down: 123
left: 89
right: 72
[INFO] [1679162317.920212]: new state: south
[INFO] [1679162322.923963]: previous state: south
[INFO] [1679162322.924810]: received data in controller:
up: 183
down: 58
left: 185
right: 115
[INFO] [1679162322.925541]: new state: south
mahdi@mahdi: ~/Desktop/Robotics/project1/hw1_ws
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ . devel/setup.bash
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ rosrun my_turtle motor1.py
[INFO] [1679162317.922989]: received data in motor1
direction: "clockwise"
value: 180
[INFO] [1679162322.927024]: received data in motor1
direction: "dont rotate"
value: 0
[INFO] [1679162327.920401]: received data in motor1
direction: "clockwise"
value: 180
[INFO] [1679162332.926520]:
mahdi@mahdi: ~/Desktop/Robotics/project1/hw1_ws
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ . devel/setup.bash
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ rosrun my_turtle motor2.py
[INFO] [1679162317.921372]: received data in motor2
direction: "clockwise"
value: 180
[INFO] [1679162322.926005]: received data in motor2
direction: "dont rotate"
value: 0
[INFO] [1679162327.927810]: received data in motor2
direction: "clockwise"
value: 180
[INFO] [1679162332.926444]:
mahdi@mahdi: ~/Desktop/Robotics/project1/hw1_ws
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ . devel/setup.bash
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ rosrun my_turtle sensor.py
[INFO] [1679162332.922410]: published data from sensor:
up: 198
down: 38
left: 187
right: 43
[INFO] [1679162337.922403]: published data from sensor:
up: 101
down: 59
left: 53
right: 118
[INFO] [1679162342.922385]: published data from sensor:
up: 135
down: 191
left: 150
right: 90
[INFO] [1679162347.922378]: published data from sensor:
value: 180
[INFO] [1679162332.926520]: received data in motor1
direction: "dont rotate"
value: 0
[INFO] [1679162337.926597]: received data in motor1
direction: "clockwise"
value: 90
[INFO] [1679162342.928463]: received data in motor1
direction: "counter clockwise"
value: 90
[INFO] [1679162347.927874]: received data in motor1
value: 90
mahdi@mahdi: ~/Desktop/Robotics/project1/hw1_ws
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ . devel/setup.bash
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ rosrun my_turtle controller.py
[INFO] [1679162332.923866]: previous state: north
[INFO] [1679162332.924718]: received data in controller:
up: 198
down: 38
left: 187
right: 43
[INFO] [1679162332.925445]: new state: north
[INFO] [1679162337.923763]: previous state: north
[INFO] [1679162337.924518]: received data in controller:
up: 101
down: 59
left: 53
right: 118
[INFO] [1679162337.924973]: new state: east
[INFO] [1679162342.924387]: previous state: east
[INFO] [1679162342.926381]: received data in controller:
up: 135
down: 191
left: 150
right: 90
[INFO] [1679162342.926960]: new state: north
[INFO] [1679162332.926444]: received data in motor2
direction: "dont rotate"
value: 0
[INFO] [1679162337.926502]: received data in motor2
direction: "clockwise"
value: 90
[INFO] [1679162342.927945]: received data in motor2
direction: "counter clockwise"
value: 90
```

- برای مثال در شماره ۱، جهت اولیه شمال بوده ولی دیدیم که نزدیکترین مانع در روبروی ما (up) بوده پس ۱۸۰ درجه (در این حالت ساعتگرد و پاد ساعتگرد فرقی ندارد) چرخیده است و state = south شده است.
- در حالت شماره ۲، ما به سمت جنوب میرفتیم و در این حالت متوجه شدیم مانع نزدیک در پشت ما (down) بوده پس نباید چرخشی اتفاق افتد (don't rotate) و مقدار چرخش هم طبیعتاً خواهد بود.
- در حالت شماره ۳، ما به سمت شمال میرفتیم و دیدیم سمت چپ(left) ما نزدیکترین مانع قرار دارد پس ۹۰ درجه ساعتگرد چرخیدیم و بعد به سمت شرق در ادامه میرویم.
- در حالت شماره ۴، به سمت شرق میرفتیم که دیدیم سمت راست (right) نزدیکترین مانع است پس ۹۰ درجه پاد ساعتگرد چرخیدیم و به سمت شمال راه را پیش گرفتیم.

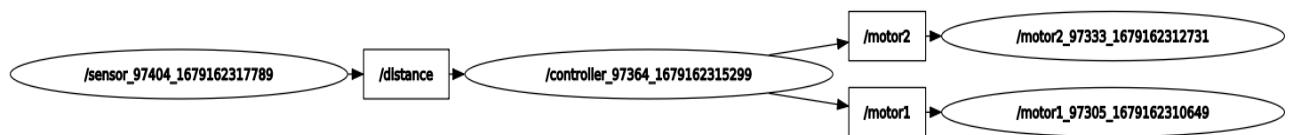
همچنین اگر rqt_graph را بخواهیم رسم کنیم لازم است تا یک ترمینال دیگر باز کنیم :



```
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ cd Desktop/Robotics/project1/hw1_ws/
mahdi@mahdi:~/Desktop/Robotics/project1/hw1_ws$ rqt_graph
```



به طور واضح تر در زیر آمده است:



اگر در کدهای مربوط به نودها anonymous = True را نگذاریم این پسوندھای شماره که به صورت رندوم تخصیص داده شدن حذف میشوند. تصویر آن در زیر آمده است.

