به نام خدا

**دانشگاه صنعتی امیرکبیر**

**دانشکده مهندسی کامپیوتر**

# اصول علم ربات

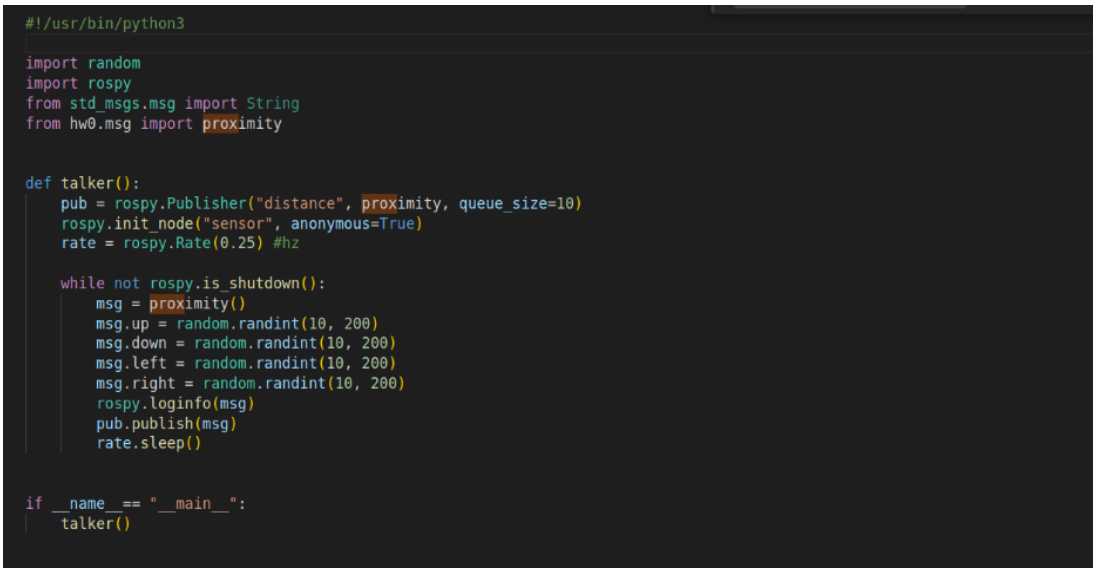
**تمرین سری اول**

| | |
|---|---|
| نام و نام خانوادگی | شایان بالی |
| شماره دانشجویی | ۹۸۳۱۰۱۴ |
| تاریخ ارسال گزارش | ۱۳ فروردین ۱۴۰۲ |

## فهرست گزارش سوالات (لطفاً پس از تکمیل گزارش، این فهرست را به‌روز کنید.)

در این تمرین قصد داشتیم که با اصول اولیه برنامه نویسی ROS آشنا بشویم و در ادامه با Node های مختلف و وظایف آن آشنا می‌شویم.

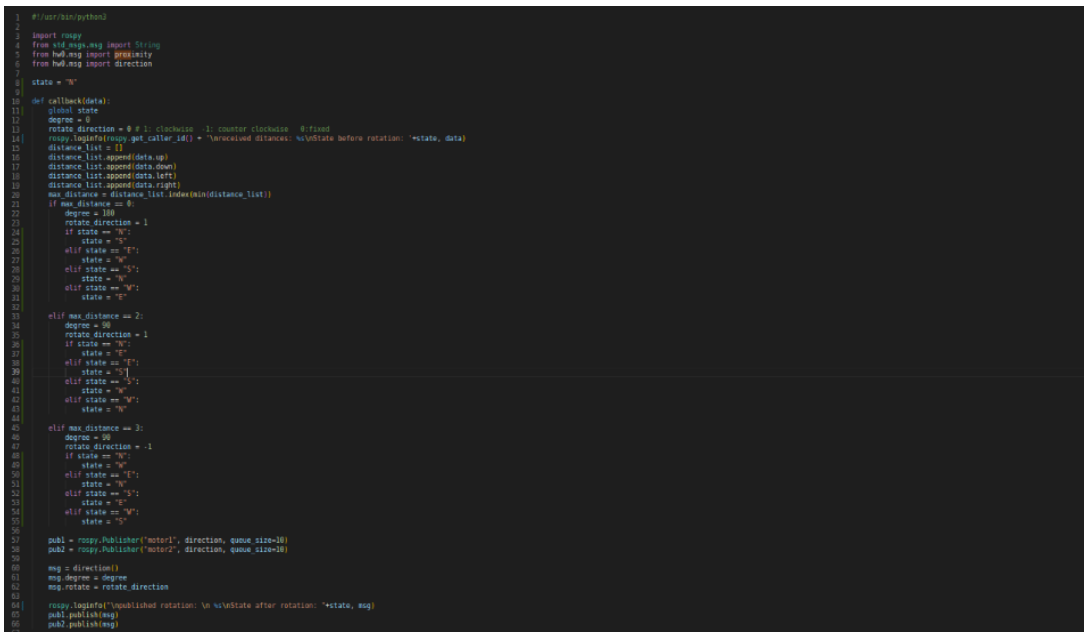
**Sensor:**



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

import random
import rospy
from std_msgs.msg import String
from hw0.msg import proximity


def talker():
    pub = rospy.Publisher("distance", proximity, queue_size=10)
    rospy.init_node("sensor", anonymous=True)
    rate = rospy.Rate(0.25) #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()
```

همانطور که مشخص است، این گره اعداد رندومی برای فاصله از ۴ جهت در بازه‌ی ۱۰ تا ۲۰۰ ایجاد می‌کند و براساس این فواصل مسیج مربوطه را ایجاد کرده و تاپیک مربوط به آن را پابلیش می‌کند.

**Controller:**



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

import rospy
from std_msgs.msg import String
from hw0.msg import proximity
from hw0.msg import direction

state = "N"

def callback(data):
    global state
    degree = 0
    rotate_direction = 0 # 1: clockwise  -1: counter clockwise  0:fixed
    rospy.loginfo(rospy.get_caller_id() + "\nreceived distances: %s\nState before rotation: "+state, data)
    distance_list = []
    distance_list.append(data.up)
    distance_list.append(data.down)
    distance_list.append(data.left)
    distance_list.append(data.right)
    max_distance = distance_list.index(min(distance_list))
    if max_distance == 0:
        degree = 180
        rotate_direction = 1
        if state == "N":
            state = "S"
        elif state == "S":
            state = "N"
        elif state == "E":
            state = "W"
        elif state == "W":
            state = "E"

    elif max_distance == 2:
        degree = 90
        rotate_direction = 1
        if state == "N":
            state = "E"
        elif state == "E":
            state = "S"
        elif state == "S":
            state = "W"
        elif state == "W":
            state = "N"

    elif max_distance == 3:
        degree = 90
        rotate_direction = -1
        if state == "N":
            state = "W"
        elif state == "E":
            state = "N"
        elif state == "S":
            state = "E"
        elif state == "W":
            state = "S"

    pub1 = rospy.Publisher("motor1", direction, queue_size=10)
    pub2 = rospy.Publisher("motor2", direction, queue_size=10)

    msg = direction()
    msg.degree = degree
    msg.rotate = rotate_direction

    rospy.loginfo("\npublished rotation: \n %s\nState after rotation: "+state, msg)
    pub1.publish(msg)
    pub2.publish(msg)
```

این گره نیز وظیفه مشخص کردن دستورات کنترلی را دارد. در ابتدای این بخش ما یک متغییر داریم جهت تعیین state ربات که در ابتدا به سمت شمال است و بعد از تعیین دستورات کنترلی این state نیز آپدیت می‌شود. در این گره نیز ابتدا تاپیک مربوط به فاصله را سابسکرایب میکند و مسیج مربوط به آن را دریافت می‌کند تا دستورات کنترلی را تعیین کند که شامل زاویه چرخش و جهت چرخش آن است که یک یعنی ساعتگرد و منفی یک یعنی پادساعتگرد و صفر نیز ثابت است. در نهایت نیز برای هریک از موتورها یک تاپیک ایجاد میکنیم تا مسیج را برای هریک از موتورها پابلیش کند.

**Motor**:

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

import rospy
from hw0.msg import direction

def callback(data):
    rt = "clockwise"
    if data.rotate == -1:
        rt = "counter clockwise"
    elif data.rotate == 0:
        rt = "fixed"
    rospy.loginfo(rospy.get_caller_id() + '\n%s ' + rt, data)

def listener():

    rospy.init_node('motor11', anonymous=True)

    rospy.Subscriber('motor1', direction, callback)

    # spin() simply keeps python from exiting until this node is stopped
    rospy.spin()

if __name__ == '__main__':
    listener()
```

برای گره های مربوط به موتورها نیز کد مشابه است و فقط تاپیکی که سابسکرایب میکنند متفاوت است؛ به این صورت که در هر گره موتور تاپیک مرتبطه سابسکرایب میشود و اطلاعات داخل مسیج را لاگ میکند.

**نمونه از نتایج:**

در ابتدا میبینیم که سنسور چهار عدد رندوم در بازه ۱۰ تا ۲۰۰ ایجاد میکند و دو دور اجرا از لوپ این گره را میبینیم.

```
[INFO] [1680481289.157927]: up: 110
down: 153
left: 167
right: 104
[INFO] [1680481293.156199]: up: 135
down: 27
left: 199
right: 88
```

همانطور که قابل مشاهده است مسیج های سابسکرایب شده چاپ گردیدند و براساس فاصله چرخش مناسب تعیین میگردد و پابلیش میشود و براساس آن state ربات نیز که جهت آن است آپدیت میشود.

```
left: 167
right: 104
State before rotation: N
[INFO] [1680481289.160836]:
published rotation:
 degree: 90
rotate: -1
State after rotation: W
[INFO] [1680481293.158098]: /controller_198329_1680479594418
received ditances: up: 135
down: 27
left: 199
right: 88
State before rotation: W
[INFO] [1680481293.159404]:
published rotation:
 degree: 0
rotate: 0
State after rotation: W
```

در ادامه هم اطلاعات لاگ شده در موتور ۱ و ۲ را میبنیم و همانطور که قابل مشاهده است وقتی فاصله نسبت به پشت ربات کمترین است، چرخشی نداریم و فیکس است.

```
[INFO] [1680481289.161980]: /motor11_149715_168047088
5846
degree: 90
rotate: -1 counter clockwise
[INFO] [1680481293.160781]: /motor11_149715_168047088
5846
degree: 0
rotate: 0 fixed
```

```
[INFO] [1680481289.162007]: /motor22_149764_1680470891351
degree: 90
rotate: -1 counter clockwise
[INFO] [1680481293.160792]: /motor22_149764_1680470891351
degree: 0
rotate: 0 fixed
```

در ادامه هم گراف مربوطه که با دستور rqt_graph ایجاد شده است را مشاهده می‌کنیم که ساختاری همانند گراف دستور کار دارد.