دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)

درس رباتیک
گزارش تمرین صفر

نگارش
محمدرضا حیدری : **9926053**

استاد درس
دکتر جوانمردی

فرودین ماه **1403**

فهرست

منطق تمرین

ربات ابتدا نزدیک ترین مانع به خود را با استفاده از داده سنسور ها تشخیص داده و سپس در جهتی چرخش کند تا به سمت مانع دور تر باشد و به سمت آن حرکت کند تا زمانی که از مانع نزدیک تر 10 سانتی متر فاصله بگیرد.

```
            {
    "1": "57 86 95 180",
    "2": "58 87 92 175",
    "3": "59 89 91 174",
    "4": "60 92 80 173",
    "5": "61 94 80 172",
    "6": "62 95 75 171",
    "7": "65 96 74 165",
    "8": "79 101 73 150",
    "9": "80 103 57 145",
    "10": "90 110 43 140"
}
```

در اولین فراخوانی کوچکترین داده 57 است و بزرگترین 180 پس باید به سمت داده بزرگتر بچرخیم و ادامه دهیم.
در دومین فراخوانی از 57 ، 10 سانتی متر فاصله نگرفته ایم و باید ادامه دهیم این روند تا عدد 79 ادامه دارد که فاصله از 57 از ده سانتی متر بیشتر است اکنون نزدیک ترین مانع 73 است و داده بزرگتر 150 و دوباره روند قبلی طی می شود

پیام ها و تعریف هر یک

1) پیام proximity.msg فاصله مانع ها تا ربات را بیان می کند.

```
int64 left
int64 up
int64 right
int64 down
```

2) پیام order.msg دستورات ارسالی به موتور را مشخص میکند .

```
int64 rotate_robot_degree
string rotate_wheel
int64 velocity
string rotate_robot_direction
```

rotate_robot_degree : مقدار چرخش ربات به درجه
rotate_wheel : جهت چرخش چرخ ، منفی به معنای عقب گرد و مثبت به معنای حرکت به جلو است.
Velocity : بیانگر سرعت خطی است که واحد آن سانتی متر بر ثانیه است.
rotate_robot_direction : بیانگر جهت چرخش کل ربات است که یا ساعتگرد هست یا پاد ساعتگرد.

سنسور

وظیفه فایل و node سنسور ، خواندن اطلاعات فایل distances.json در هر فراخوانی است و درنهایت باید اطلاعات را در قالب یک پیام شخصی سازی شده publish کند.

فایل سنسور دارای سه تابع است که وظایف هر کدام به شرح زیر است :

1) distance_sensor_node_define

تعریف نود سنسور و تعریف یک تاپیک به نام distance

2) read_distance

خواندن اطلاعات از فایل distances.json

3) start

این تابع اصلی است که کارهای سنسور را هماهنگ کرده و از دو تابع قبلی استفاده میکند نهایتا پیام را publish می کند.

```
#!/usr/bin/python3
import rospy
from hwzero.msg import proximity
import json
import os
iteration = 1
def read_distance(iteration):
    script_dir = os.path.dirname(__file__)
    file_path = os.path.join(script_dir, 'distances.json')
    with open(file_path, 'r') as f:
        distance_data = json.load(f)
    return [int(x) for x in distance_data[str(iteration)].split()]

def distance_sensor_node_define():
    rospy.init_node("sensor",anonymous=True)
    publisher = rospy.Publisher("distance",proximity,queue_size=10)
    rate = rospy.Rate(0.75)
    return publisher,rate
def start():
 publisher,rate = distance_sensor_node_define()
 global iteration
 while not rospy.is_shutdown() and iteration<=10:
    rate.sleep()
    rospy.loginfo(iteration)
    left, up, right, down = read_distance(iteration)
    distance_msg = proximity()
    distance_msg.left = left
    distance_msg.up = up
    distance_msg.right = right
    distance_msg.down = down
```

```
    publisher.publish(distance_msg)
    iteration = iteration+1
if __name__ == '__main__':
    start()
```

خروجی سنسور

هر عدد بیانگر تعداد دفعات خواندن اطلاعات از فایل json است.

```
heydari@MohammadReza:~$ rosrun hwzero sensor.py
[INFO] [1712080126.491010]: 1
[INFO] [1712080127.824595]: 2
[INFO] [1712080129.157718]: 3
[INFO] [1712080130.491335]: 4
[INFO] [1712080131.824382]: 5
[INFO] [1712080133.157724]: 6
[INFO] [1712080134.491184]: 7
[INFO] [1712080135.824355]: 8
[INFO] [1712080137.157838]: 9
[INFO] [1712080138.491650]: 10
```

کنترلر

وظیفه فایل و نود کنترلر تصمیم گیری بر اساس اطلاعات سنسور و فرستادن دستور مناسب به هر یک از موتور ها است

فایل کنترلر دارای پنج تابع اصلی است که وظایف هر یک به شرح زیر است :

1) callback

آماده سازی اطلاعات ارسالی از سنسور برای استفاده دیگر توابع

2) controller_node

تعریف نود کنترلر و دو تاپیک برای دستور به موتور ها همچنین تعریف یک دریافت کننده اطلاعات از سنسور

3) create_order_msg

آماده سازی پیام مناسب جهت ارسال به موتور ها ، قسمت زیادی از منطق برنامه در این تابع است.

4) create_zero_messaage

آماده سازی پیام مناسب به کنترلر وقتی نیازی به چرخش ربات نیست و ربات باید به جهت حرکت خود ادامه دهد.

5) start2

این تابع اصلی است که کارهای کنترلر را هماهنگ کرده و از چهار تابع قبلی استفاده میکند نهایتا پیام ها را publish می کند.قسمتی از منطق های برنامه نیز در این تابع به طور جداگانه برنامه ریزی شده است.

```
#!/usr/bin/python3
import rospy
from hwzero.msg import proximity
from hwzero.msg import order
import random
```

```
import time
Data = None
Data_Old = None
def callback(data):
    global Data
    Data = data

def controller_node():
    rospy.init_node('controller', anonymous=True)
    engine1_pub = rospy.Publisher('engine1_order', order, queue_size=10)
    engine2_pub = rospy.Publisher('engine2_order', order, queue_size=10)
    rospy.Subscriber('distance', proximity, callback)
    return engine1_pub, engine2_pub

def start2():
    engine1_pub, engine2_pub = controller_node()
    while not rospy.is_shutdown():
        if Data:
            start_time = time.time()
            engine_msg1,engine_msg2 = create_order_msg(Data)
            engine_msg1.velocity = 0
            engine_msg2.velocity = 0
            engine1_pub.publish(engine_msg1)
            engine2_pub.publish(engine_msg2)
            rospy.loginfo("first data recevied")
            Data_Old = Data
            Data_List = [Data.left,Data.up,Data.down,Data.right]
            Data_old_list =
[Data_Old.left,Data_Old.up,Data_Old.down,Data_Old.right]
            min_number = min(Data_old_list)
            min_index = Data_old_list.index(min_number)
            min_element = Data_List[min_index]
            while not rospy.is_shutdown():
                Data_List = [Data.left,Data.up,Data.down,Data.right]
                Data_old_list =
[Data_Old.left,Data_Old.up,Data_Old.down,Data_Old.right]
                min_number = min(Data_old_list)
                min_index = Data_old_list.index(min_number)
                min_element = Data_List[min_index]
                if max(Data_List) == 140 and max(Data_old_list)==140:
                    rospy.signal_shutdown("Shutting down")
                    exit()
                rospy.loginfo('min of old distance : %s' , min_number )
                rospy.loginfo('now we are in  : %s' , min_element )
```

```
                if min_element - min_number > 10 or min_element - min_number < -
10 :
                    engine_msg1,engine_msg2 = create_order_msg(Data)
                    elapsed_time = time.time() - start_time
                    engine_msg1.velocity = abs(int((min_element -
min_number)/elapsed_time))
                    engine_msg2.velocity = engine_msg1.velocity
                    start_time = time.time()
                    rospy.loginfo("we should rotate")
                    engine1_pub.publish(engine_msg1)
                    engine2_pub.publish(engine_msg2)
                    Data_Old = Data
                else:
                    engine_msg1,engine_msg2 = create_zero_messaage()
                    rospy.loginfo("do not need rotate")
                    elapsed_time = time.time() - start_time
                    engine_msg1.velocity = abs(int((min_element -
min_number)/elapsed_time))
                    engine_msg2.velocity = engine_msg1.velocity
                    start_time = time.time()
                    engine1_pub.publish(engine_msg1)
                    engine2_pub.publish(engine_msg2)
                rate = rospy.Rate(.75)
                rate.sleep()

def create_order_msg(Data):
    Data2 = [Data.left,Data.up,Data.down,Data.right]
    def find_way(Data):
        if max(Data2) == Data.left:
            maxIndex = "left"
        if max(Data2) == Data.right:
            maxIndex = "right"
        if max(Data2) == Data.up:
            maxIndex = "up"
        if max(Data2) == Data.down:
            maxIndex = "down"


        if min(Data2) == Data.left:
            minIndex = "left"
        if min(Data2) == Data.right:
            minIndex = "right"
        if min(Data2) == Data.up:
            minIndex = "up"
        if min(Data2) == Data.down:
```

```
            minIndex = "down"


        if maxIndex == "left":
            if minIndex == "right":
                way = 2
                sgn = 1
            if minIndex == "down":
                way = 1
                sgn = 1
            if minIndex == "up":
                way = 1
                sgn = -1
        if maxIndex == "right":
            if minIndex == "left":
                way = 2
                sgn = 1
            if minIndex == "down":
                way = 1
                sgn = -1
            if minIndex == "up":
                way = 1
                sgn = 1
        if maxIndex == "down":
            if minIndex == "right":
                way = 1
                sgn = 1
            if minIndex == "left":
                way = 1
                sgn = -1
            if minIndex == "up":
                way = 2
                sgn = 1
        if maxIndex == "up":
            if minIndex == "right":
                way = 1
                sgn = -1
            if minIndex == "down":
                way = 2
                sgn = 1
            if minIndex == "left":
                way = 1
                sgn = 1
        return way,sgn
    way , sgn = find_way(Data)
```

```
    def UpLeft_LeftDown_DownRight_RightUp(sgn):
        engine_msg1 = order()
        engine_msg2 = order()
        if sgn == 1:
            rospy.loginfo("robot should rotate 90 degree ccw")
            engine_msg1.rotate_robot_degree = 90
            engine_msg1.rotate_wheel = "negative"
            engine_msg1.velocity = None
            engine_msg1.rotate_robot_direction = "ccw"
            engine_msg2.rotate_robot_degree = 90
            engine_msg2.rotate_wheel = "positive"
            engine_msg2.velocity = None
            engine_msg2.rotate_robot_direction = "ccw"
        if sgn == -1:
            rospy.loginfo("robot should rotate 90 degree cw")
            engine_msg1.rotate_robot_degree = 90
            engine_msg1.rotate_wheel = "positive"
            engine_msg1.velocity = None
            engine_msg1.rotate_robot_direction = "cw"
            engine_msg2.rotate_robot_degree = 90
            engine_msg2.rotate_wheel = "negative"
            engine_msg2.velocity = None
            engine_msg2.rotate_robot_direction = "cw"
        return engine_msg1,engine_msg2
    def UpDown_LeftRight(sgn):
        sgn = random.choice([1, -1])
        engine_msg1 = order()
        engine_msg2 = order()
        rospy.loginfo("robot should rotate 180 degree cw or ccw")
        if sgn == 1:
            engine_msg1.rotate_robot_degree = 180
            engine_msg1.rotate_wheel = "positive"
            engine_msg1.velocity = None
            engine_msg1.rotate_robot_direction = "cw"
            engine_msg2.rotate_robot_degree = 180
            engine_msg2.rotate_wheel = "negative"
            engine_msg2.velocity = None
            engine_msg2.rotate_robot_direction = "cw"
        if sgn == -1:
            engine_msg1.rotate_robot_degree = 180
            engine_msg1.rotate_wheel = "negative"
            engine_msg1.velocity = None
            engine_msg1.rotate_robot_direction = "ccw"
            engine_msg2.rotate_robot_degree = 180
            engine_msg2.rotate_wheel = "positive"
```

```
            engine_msg2.velocity = None
            engine_msg2.rotate_robot_direction = "ccw"

        return engine_msg1,engine_msg2
    if way == 1:
        engine_msg1,engine_msg2 = UpLeft_LeftDown_DownRight_RightUp(sgn)
    if way == 2:
        engine_msg1,engine_msg2 = UpDown_LeftRight(sgn)
    return engine_msg1,engine_msg2
def create_zero_messaage():
    engine_msg1 = order()
    engine_msg2 = order()
    engine_msg1.rotate_robot_degree = 0
    engine_msg1.rotate_wheel = "positive"
    engine_msg1.velocity = None
    engine_msg1.rotate_robot_direction = 0
    engine_msg2.rotate_robot_degree = 0
    engine_msg2.rotate_wheel = "positive"
    engine_msg2.velocity = None
    engine_msg2.rotate_robot_direction = "None"
    return engine_msg1,engine_msg2


if __name__ == '__main__':
    start2()
```

نکات تکمیلی کد کنترلر :

- متغیر way در صورت برابر بودن با عدد یک یعنی یک واحد چرخش نیاز است و برابر بودن با عدد دو یعنی دو واحد یا 180 در جه چرخش نیاز است.
- متغیر sgn در صورت برابر بودن با یک یعنی پاد ساعتگرد و برابر بودن با منفی یک یعنی ساعتگرد
- سرعت های لحظه ای در موتور نمایش داده می شود که در دیتای دریافتی اول صفر نشان داده می شود.

خروجی کنترلر

```
heydari@MohammadReza:~$ source catkin2_ws/devel/setup.bash
heydari@MohammadReza:~$ rosrun hwzero controller.py
[INFO] [1712080126.497999]: robot should rotate 90 degree cw
[INFO] [1712080126.499141]: first data recevied
[INFO] [1712080126.500000]: min of old distance : 57
[INFO] [1712080126.501392]: now we are in  : 57
[INFO] [1712080126.502135]: do not need rotate
[INFO] [1712080127.838347]: min of old distance : 57
[INFO] [1712080127.839418]: now we are in  : 58
[INFO] [1712080127.840140]: do not need rotate
[INFO] [1712080129.175945]: min of old distance : 57
[INFO] [1712080129.177104]: now we are in  : 59
[INFO] [1712080129.177934]: do not need rotate
[INFO] [1712080130.513716]: min of old distance : 57
[INFO] [1712080130.514838]: now we are in  : 60
[INFO] [1712080130.515861]: do not need rotate
[INFO] [1712080131.852062]: min of old distance : 57
[INFO] [1712080131.853153]: now we are in  : 61
[INFO] [1712080131.853837]: do not need rotate
[INFO] [1712080133.190103]: min of old distance : 57
[INFO] [1712080133.193186]: now we are in  : 62
[INFO] [1712080133.194245]: do not need rotate
[INFO] [1712080134.530291]: min of old distance : 57
[INFO] [1712080134.531283]: now we are in  : 65
[INFO] [1712080134.532068]: do not need rotate
[INFO] [1712080135.868350]: min of old distance : 57
[INFO] [1712080135.872833]: now we are in  : 79
[INFO] [1712080135.877047]: robot should rotate 90 degree ccw
[INFO] [1712080135.878120]: we should rotate
[INFO] [1712080137.214445]: min of old distance : 73
[INFO] [1712080137.215815]: now we are in  : 57
[INFO] [1712080137.216850]: robot should rotate 90 degree ccw
[INFO] [1712080137.217708]: we should rotate
[INFO] [1712080138.553852]: min of old distance : 57
[INFO] [1712080138.555273]: now we are in  : 43
[INFO] [1712080138.556150]: robot should rotate 90 degree ccw
[INFO] [1712080138.557086]: we should rotate
heydari@MohammadReza:~$
```

موتور ها

وظیفه دو فایل مربوط به موتور فقط تعریف دو نود و دریافت و چاپ دستورات کنترلر است.

منظور از موتور اول موتور سمت چپ و موتور دوم موتور سمت راست است.

برای مثال فایل موتور اول به شرح زیر است :

```python
#!/usr/bin/python3
import rospy
from hwzero.msg import order

def engine_node_callback1(data):
    rospy.loginfo("engine1 :%s",data)
```

```python
def start3():
    rospy.init_node('engine_node', anonymous=True)
    rate = rospy.Rate(.75)
    while not rospy.is_shutdown():
        rate.sleep()
        rospy.Subscriber('engine1_order', order, engine_node_callback1)

if __name__ == '__main__':
    start3()
```

قسمتی از خروجی موتور یک

```
[INFO] [1712083644.170222]: engine1 :rotate_robot_degree: 0
rotate_wheel: "positive"
velocity: 5
rotate_robot_direction: "None"
[INFO] [1712083645.509144]: engine1 :rotate_robot_degree: 90
rotate_wheel: "negative"
velocity: 16
rotate_robot_direction: "ccw"
[INFO] [1712083645.516690]: engine1 :rotate_robot_degree: 90
rotate_wheel: "negative"
velocity: 16
rotate_robot_direction: "ccw"
[INFO] [1712083645.518930]: engine1 :rotate_robot_degree: 90
rotate_wheel: "negative"
velocity: 16
rotate_robot_direction: "ccw"
[INFO] [1712083645.520736]: engine1 :rotate_robot_degree: 90
rotate_wheel: "negative"
velocity: 16
```

قسمتی از خروجی موتور دو

```
[INFO] [1712083644.165043]: engine2 :rotate_robot_degree: 0
rotate_wheel: "positive"
velocity: 5
rotate_robot_direction: "None"
[INFO] [1712083644.166243]: engine2 :rotate_robot_degree: 0
rotate_wheel: "positive"
velocity: 5
rotate_robot_direction: "None"
[INFO] [1712083645.508894]: engine2 :rotate_robot_degree: 90
rotate_wheel: "positive"
velocity: 16
rotate_robot_direction: "ccw"
[INFO] [1712083645.515836]: engine2 :rotate_robot_degree: 90
rotate_wheel: "positive"
velocity: 16
rotate_robot_direction: "ccw"
[INFO] [1712083645.518059]: engine2 :rotate_robot_degree: 90
rotate_wheel: "positive"
velocity: 16
rotate_robot_direction: "ccw"
[INFO] [1712083645.519973]: engine2 :rotate_robot_degree: 90
rotate_wheel: "positive"
velocity: 16
```

راهنمای اجرای کد

آخرین فایلی که اجرا می شود باید sensor باشد چون sensor به محض راه اندازی دیتا ارسال میکند و به خاط کم بودن دیتا ها این باعث می شود بعد از ده ثانیه اطلاعات تمام شود پس حتما اخرین فایلی که اجرا میکنید باید sensor باشد.

ترتیب اجرای دیگر فایل ها مهم نیست.

Rqt_graph