チャレンジサイト・メカニックカモノハシ 2019
マイクロマウスシミュレータ Exercise

ER17045 立道壱太郎

2019 年 7 月 14 日

1　この資料について

メカニックカモノハシではマイクロマウス大会に参加することでメンバーの技術向上を図ります。

2　マイクロマウスパッケージの導入

2.1　workspace の作成

メカニックカモノハシ用の workspace を新たに作りましょう。

ソースコード 1　workspace の作成

```
mkdir −p ~/mp_ws/src
cd ~/mp_ws/src/
catkin_init_workspace
echo ”source ~/mp_ws/devel/setup.bash” >> ~/.bashrc
```

2.2　マイクロマウスパッケージの導入

github から、パッケージを clone して catkin_make します。

ソースコード 2　catkin_make

```
cd ~/mp_ws/src
git clone
cd ~/mp_ws
catkin_ws
source ~/.bashrc
```

以下のコマンドを入力し、ディレクトリを移動できれば成功です。

ソースコード 3　roscd

```
roscd micro_mouse
```

3 関連パッケージの導入

マイクロマウスパッケージの動作に必要な関連パッケージを導入します。以下のコマンドを実行してください。(全部で一行)

ソースコード 4　roscd

```
sudo apt install ros-kinetic-turtlebot ros-kinetic-turtlebot-msgs ros-kinetic-turtlebot-
    teleop
```

4 Exercise

4.1 Python をマスターしよう

4.2 マウスを動かそう

ソースコード 5　micro_mouse/script/chapter/1/moving_mouse.py

```
1  #!/usr/bin/env python
2  #coding: utf−8 # 日本語を使えるようにする
3
4  import roslib.packages
5  pk_path = roslib.packages.get_pkg_dir('micro_mouse')
6  import sys
7  sys.path.append(pk_path + '/script/lib')
8
9  import math
10 import rospy # include<ros/ros.h>のようなもの
11
12 import mouse
13
14
15 if __name__ == '__main__': # int main()みたいな
16   rospy.init_node("node_name") # ノード設定
17
18   mouse = mouse.Mouse( "cmd_vel", ["scan/R",
19                                    "scan/FR",
20                                    "scan/F",
21                                    "scan/FL",
22                                    "scan/L"])
23   try:
24     loop = rospy.Rate(10) # 10[loop/s]の設定をする。
25     while not rospy.is_shutdown():
26 #−−−−−ここにプログラムを書く−−−−−
27   #−−−−−ロボットを動かす−−−−−
28       # mouse.move( 直進速度 [ m/s], 回転速度 [ rad/s])
29       # 直進速度は 1[m/s]、回転速度は π /2 [ rad/s]が限界です。
30       mouse.move( 1, 0.0)
31       mouse.move(0.0, 2 ∗ math.pi/2)
32       mouse.move( 1, 0.0)
33       mouse.move( −1, 0.0)
34       mouse.move(0.0, −1 ∗ math.pi/2)
35       mouse.move( −1, 0.0)
36
37 #−−−−−ここまで−−−−−
38       loop.sleep() # 10[loop/s]になるよう調整する。
39   except KeyboardInterrupt: # 実行中 (try:中)にCTRL−C が押されればプログラムが終了する
40     print("キーボード割り込み！CTRL−C　終了")
```

ソースコード 6　micro_mouse/script/chapter/1/going_mouse.py

```
1  #!/usr/bin/env python
2  #coding: utf−8 # 日本語を使えるようにする
3
4  import roslib.packages
5  pk_path = roslib.packages.get_pkg_dir('micro_mouse')
6  import sys
7  sys.path.append(pk_path + '/script/lib')
8
9  import math
10 import rospy # include<ros/ros.h>のようなもの
11
12 import mouse
13
14
15 if __name__ == '__main__': # int main()みたいな
16   rospy.init_node("node_name") # ノード設定
17
18   mouse = mouse.Mouse( "cmd_vel", ["scan/R",
19                                    "scan/FR",
20                                    "scan/F",
21                                    "scan/FL",
22                                    "scan/L"])
23   try:
24     loop = rospy.Rate(10) # 10[loop/s]の設定をする。
25     while not rospy.is_shutdown():
26 #−−−−−ここにプログラムを書く−−−−−
27   #−−−−−ロボットを動かす−−−−−
28       # mouse.move( 直進速度 [ m/s], 回転速度 [ rad/s])
29       # 直進速度は 1[m/s]、回転速度は π /2 [ rad/s]が限界です。
30       mouse.move( 1, 0.0)
31       mouse.move(0.0, 2 ∗ math.pi/2)
32       mouse.move( 1, 0.0)
33       mouse.move( −1, 0.0)
34       mouse.move(0.0, −1 ∗ math.pi/2)
35       mouse.move( −1, 0.0)
36
37 #−−−−−ここまで−−−−−
38       loop.sleep() # 10[loop/s]になるよう調整する。
39   except KeyboardInterrupt: # 実行中 (try:中)にCTRL−C が押されればプログラムが終了する
40     print("キーボード割り込み！CTRL−C　終了")
```

ソースコード 7　micro_mouse/script/chapter/1/sensing_mouse.py

```
#!/usr/bin/env python
#coding: utf−8 # 日本語を使えるようにする

import roslib.packages
pk_path = roslib.packages.get_pkg_dir('micro_mouse')
import sys
sys.path.append(pk_path + '/script/lib')

import math
import rospy # include<ros/ros.h>のようなもの

import mouse


if __name__ == '__main__': # int main()みたいな
  rospy.init_node("node_name") # ノード設定

  mouse = mouse.Mouse( "cmd_vel", ["scan/R",
                                   "scan/FR",
                                   "scan/F",
                                   "scan/FL",
                                   "scan/L"])
  try:
    loop = rospy.Rate(10) # 10[loop/s]の設定をする。
    while not rospy.is_shutdown():
#−−−−−ここにプログラムを書く−−−−−
  #−−−−−ロボットを動かす−−−−−
      # mouse.move( 直進速度 [ m/s], 回転速度 [ rad/s])
      # 直進速度は 1[m/s]、回転速度は π /2 [ rad/s]が限界です。
      mouse.move( 1, 0.0)
      mouse.move(0.0, 2 * math.pi/2)
      mouse.move( 1, 0.0)
      mouse.move( −1, 0.0)
      mouse.move(0.0, −1 * math.pi/2)
      mouse.move( −1, 0.0)

#−−−−−ここまで−−−−−
      loop.sleep() # 10[loop/s]になるよう調整する。
  except KeyboardInterrupt: # 実行中 (try:中)にCTRL−C が押されればプログラムが終了する
    print("キーボード割り込み！CTRL−C　終了")
```

4.3 wallPublisher を使いこなそう

4.4 pathPublisher を使いこなそう

4.5 左手法・拡張左手法・足立法を理解しよう

ソースコード 8　micro_mouse/script/chapter/3/left_hund.py

```
#!/usr/bin/env python
#coding: utf−8 # 日本語を使えるようにする

import roslib.packages
pk_path = roslib.packages.get_pkg_dir('micro_mouse')
import sys
sys.path.append(pk_path + '/script/lib')

import math
import numpy as np
import rospy # include<ros/ros.h>のようなもの
from std_msgs.msg import Float32
from geometry_msgs.msg import Twist

import wall
import mouse


if __name__ == '__main__': # int main()みたいな
  rospy.init_node("node_name") # ノード設定

  use_sensor_num = [ 4, 2, 0]
  sensor = [ 0 for i in range( 3)]
  sensor_dire = [ 1, 0,−1]

  cell_matrix = [ 17, 17]
  cell_size = [ 0.500, 0.500]
  walls = wall.wallPublisher("wall_marker", cell_matrix, cell_size)
  mouse = mouse.Mouse( "cmd_vel", ["scan/R",
                                   "scan/FR",
                                   "scan/F",
                                   "scan/FL",
                                   "scan/L"])
  map_data = np.zeros( cell_matrix)

  vl = vr = 0
  mx = my = theta = 0
  try:
    loop = rospy.Rate(10) # 10[loop/s]の設定をする。
    while not rospy.is_shutdown():
#−−−−−ここにプログラムを書く−−−−−

  #−−−−−センサー読み込み−−−−−
      for (i, num) in enumerate(use_sensor_num):
        sensor[ i] = mouse.sensor[ num].data < 1.0
  #−−−−−検知した壁を地図に反映させる
      for (i,dire) in enumerate(sensor_dire):
        if sensor[i] == True:
          wx = int( round( mx∗2+1 + math.cos( (theta + dire) ∗ math.pi/2)))
          wy = int( round( my∗2+1 + math.sin( (theta + dire) ∗ math.pi/2)))
          map_data[ wy, wx] = 1
  #−−−−−地図情報等を表示する−−−−−
      print mx, my, theta
```

```
54          walls.setData( map_data)
55          walls.publish()
56      #−−−−−行動選択 (左手法)−−−−−
57          vl = vr = 0
58          if sensor[0] == False:
59              vl = 1; vr = 1
60          elif sensor[0] == True and sensor[1] == True:
61              vl = 0; vr = −1
62          elif sensor[0] == True and sensor[1] == False:
63              vl = 1; vr = 0
64      #−−−−−ロボットを動かす−−−−−
65          if vr != 0:
66              mouse.move(0.0, vr ∗ math.pi/2, 10)
67          if vl != 0:
68              mouse.move( vl, 0.0, 10)
69      #−−−−−ロボットの自己位置を更新する−−−−−
70          theta += vr; theta %= 4
71          mx = int( round( mx + vl ∗ math.cos( theta ∗ math.pi/2)))
72          my = int( round( my + vl ∗ math.sin( theta ∗ math.pi/2)))
73
74  #−−−−−ここまで−−−−−
75          loop.sleep() # 10[loop/s]になるよう調整する。
76      except KeyboardInterrupt: # try:中にCTRL−C が押されればココを実行する。
77          print(”キーボード割り込み！CTRL−C　終了”)
```

ソースコード 9　micro_mouse/script/chapter/3/left_hund_ex.py

```
#!/usr/bin/env python
#coding: utf−8 # 日本語を使えるようにする

import sys,os
sys.path.append(os.path.dirname(os.path.abspath(__file__)) + '/lib')

import math
import numpy as np
import rospy # include<ros/ros.h>のようなもの
from std_msgs.msg import Float32
from geometry_msgs.msg import Twist

import wall
import mouse


if __name__ == '__main__': # int main()みたいな
  rospy.init_node("node_name") # ノード設定

  use_sensor_num = [ 4, 2, 0]
  sensor = [ 0 for i in range( 4)]
  sensor_dire = [ 1, 0,−1, −2] # Left, Forward, Right, Back

  cell_matrix = [ 17, 17]
  cell_size = [ 0.500, 0.500]
  walls = wall.wallPublisher("wall_marker", cell_matrix, cell_size)
  mouse = mouse.Mouse( "cmd_vel", ["scan/R",
                                   "scan/FR",
                                   "scan/F",
                                   "scan/FL",
                                   "scan/L"])
  map_data = np.zeros( cell_matrix)
  map_data_colored = np.zeros( cell_matrix, dtype=int)

  vl = vr = 0
  mx = my = dire = 0
  try:
    loop = rospy.Rate(10) # 10[loop/s]の設定をする。
    while not rospy.is_shutdown():
#−−−−−ここにプログラムを書く−−−−−
      tx = mx∗2 + 1
      ty = my∗2 + 1
  #−−−−−センサー読み込み−−−−−
      for (i, num) in enumerate(use_sensor_num):
        sensor[ i] = mouse.sensor[ num].data < 1.0
  #−−−−−地図情報を更新する−−−−−
    #−−−−−壁更新−−−−−
      for (i,s_dire) in enumerate(sensor_dire):
        wx = int( round( tx + math.cos( (dire + s_dire) ∗ math.pi/2)))
        wy = int( round( ty + math.sin( (dire + s_dire) ∗ math.pi/2)))
        rx = int( round( tx + math.cos( (dire + s_dire) ∗ math.pi/2)∗2))
        ry = int( round( ty + math.sin( (dire + s_dire) ∗ math.pi/2)∗2))
        if sensor[i] == True:
          map_data[ wy, wx] = 1 # 壁 ( white)
        if map_data[ ty, tx] == 0 and map_data[ wy, wx] == 0:
          if not (rx < 0 or ry < 0 or cell_matrix[0] <= rx or cell_matrix[1] <= ry or i==3):
            if map_data[ ry, rx] != 0:
```

```
58                  map_data[ wy, wx] = 2 # 仮想的な壁
59            sensor[i] = (map_data[ wy, wx] != 0)
60            print i, map_data[ wy, wx]
61          print mx, my, dire
62          print "−−−−−−−−−−−−−−−−−"
63        #−−−−−道更新−−−−−
64          wall = sum( sensor)
65          if wall== 3:
66            color = 3 # 行き止まり
67          elif wall <= 1:
68            color = 4 # 交差点
69          else:
70            color = 5 # 行き止まりでも交差点でもない場所
71          map_data[ ty, tx] = color
72
73      #−−−−−マップ色変え&マップ送信−−−−−
74          map_data_colored[ map_data == 0] = walls.WALL_INVISIBLE # 何もない
75          map_data_colored[ map_data == 1] = walls.WALL_WHITE # 壁
76          map_data_colored[ map_data == 2] = walls.WALL_RED # 仮想的な壁
77          map_data_colored[ map_data == 3] = walls.WALL_GREEN # 行き止まり
78          map_data_colored[ map_data == 4] = walls.WALL_ORANGE # 交差点
79          map_data_colored[ map_data == 5] = walls.WALL_GREY # 行き止まりでも交差点でも
                ない (踏んだタイル)
80          walls.setData( map_data_colored)
81      #−−−−−行動選択 (左手法)−−−−−
82          vl = vr = 0
83          if sensor[0] == False:
84            vl = 1; vr = 1
85          elif sensor[0] == True and sensor[1] == True:
86            vl = 0; vr = −1
87          elif sensor[0] == True and sensor[1] == False:
88            vl = 1; vr = 0
89      #−−−−−地図やロボット情報を表示する−−−−−
90          walls.publish()
91      #−−−−−ロボットを動かす−−−−−
92          if vr != 0:
93            mouse.move(0.0, vr ∗ math.pi/2, 10)
94          if vl != 0:
95            mouse.move( vl, 0.0, 10)
96      #−−−−−ロボットの自己位置を更新する−−−−−
97          dire += vr; dire %= 4
98          mx = int( round( mx + vl ∗ math.cos( dire ∗ math.pi/2)))
99          my = int( round( my + vl ∗ math.sin( dire ∗ math.pi/2)))
100
101 #−−−−−ここまで−−−−−
102         loop.sleep() # 10[loop/s]になるよう調整する。
103     except KeyboardInterrupt: # try:中にCTRL−C が押されればココを実行する。
104       print("キーボード割り込み！CTRL−C　終了")
```

4.6 経路計画を理解しよう

参考文献

[1] Windows10 と Ubuntu16.04 のデュアルブート環境構築
https://qiita.com/medalotte/items/4bb5cfa709e93d044f1c

[2] Windows10 と Ubuntu18.04 をデュアルブートする.
https://qiita.com/yo_kanyukari/items/2a944a300db22482c696