

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 https://github.com/platypus5384/micro_mouse.git
$ 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 micro_mouse 動作テスト

付いていると思いますが、一応、ファイルに権限を付けます。以下を実行してください。

ソースコード 5 roscd

```
$ roscd micro_mouse/script  
$ sudo chmod -R a+x *
```

シミュレータを起動します。以下のコマンドをそれぞれ別端末で実行してください。

ソースコード 6 roscd

```
$ roslaunch micro_mouse startup.launch  
別の端末を開き、  
$ rosrn micro_mouse left_hund.ex.py
```

マウスが動き出し、マッピングが始まれば成功です。しばらく待つと、以下のようになります。

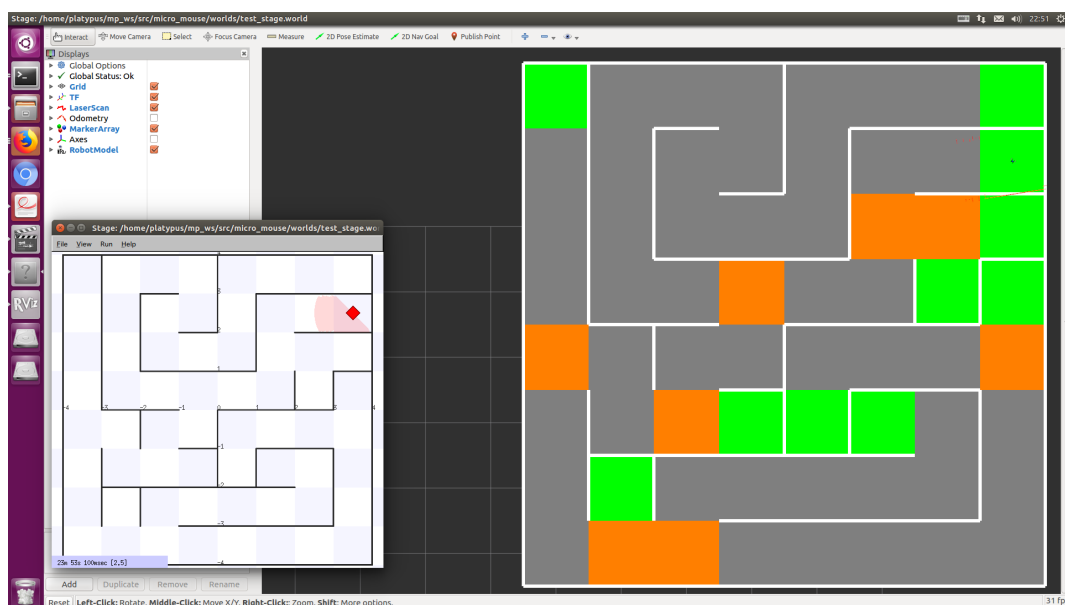


図 1

5 Exercise

5.1 Python 簡易入門

簡易的に Python 入門しよう。

5.2 マウスを動かそう

ソースコード 7 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
24     try:
25         loop = rospy.Rate(10) # 10[loop/s]の設定をする。
26         while not rospy.is_shutdown():
27             #-----ここにプログラムを書く-----
28             #-----ロボットを動かす-----
29             # mouse.move( 直進速度 [ m/s], 回転速度 [ rad/s])
30             # 直進速度は 1[m/s]、回転速度は /2 [ rad/s]が限界です。
31             mouse.move( 1, 0.0)
32             mouse.move(0.0, 2 * math.pi/2)
33             mouse.move( 1, 0.0)
34             mouse.move( -1, 0.0)
35             mouse.move(0.0, -1 * math.pi/2)
36             mouse.move( -1, 0.0)
37
38             #-----ここまで-----
39             loop.sleep() # 10[loop/s]になるよう調整する。
40     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
41         print("キーボード割り込み！CTRL-C 終了")
```

```

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 #-----変数定義-----
24 action_list = [ 1,2,2,2,3,2,3,2,1,2,1,2,3,2] # 1:左回転, 2:直進:, 3:右回転
25 vl = 0
26 vr = 0
27 #-----変数定義終了-----
28
29 try:
30     loop = rospy.Rate(10) # 10[loop/s]の設定をする。
31     while not rospy.is_shutdown():
32 #-----ここにプログラムを書く-----
33     #-----ロボットを動かす-----
34     # mouse.move( 直進速度 [ m/s], 回転速度 [ rad/s])
35     # 直進速度は 1[m/s]、回転速度は /2 [ rad/s]が限界です。
36     for action in action_list:
37         vl = 0
38         vr = 0
39         if action == 1:
40             vr = math.pi/2
41         elif action == 2:
42             vl = 1
43         elif action == 3:
44             vr = -math.pi/2
45         mouse.move( vl, vr)
46     break
47
48 #-----ここまで-----
49     loop.sleep() # 10[loop/s]になるよう調整する。
50 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
51     print("キーボード割り込み！CTRL-C 終了")

```

```

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", # 0 センサの管理番号
19                                     "scan/FR", # 1
20                                     "scan/F", # 2
21                                     "scan/FL", # 3
22                                     "scan/L"]) # 4
23 #-----変数定義-----
24 # mouse には 5 つのセンサが搭載されています。
25 # それぞれのセンサ値を管理するための配列変数sensor を定義する。
26 sensor = [0 for i in range( 5)]
27 #-----変数定義終了-----
28
29     try:
30         loop = rospy.Rate(10) # 10[loop/s]の設定をする。
31         while not rospy.is_shutdown():
32 #-----ここにプログラムを書く-----
33             #-----センサー読み込み-----
34             # mouse.sensor[管理番号]で、壁までの「距離」が取得できる。
35             # また、(距離 < 1.0)で、[距離]を [ブール値]に変える
36             for i in range(5):
37                 sensor[ i] = (mouse.sensor[ i].data < 1.0)
38 #-----ロボットを動かす-----
39             # mouse.move( 直進速度 [ m/s], 回転速度 [ rad/s])
40             # 直進速度は 1[m/s]、回転速度は /2 [ rad/s]が限界です。
41
42             if sensor[ 2] == 0:
43                 print("壁がない、前に進もう")
44                 mouse.move( 1, 0.0)
45             else:
46                 print("壁がある、左回転しよう")
47                 mouse.move( 0, math.pi/2)
48
49 #-----ここまで-----
50         loop.sleep() # 10[loop/s]になるよう調整する。
51     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
52         print("キーボード割り込み！CTRL-C 終了")

```

5.3 wallPublisher を使いこなそう

5.4 pathPublisher を使いこなそう

5.5 左手法・拡張左手法・足立法を理解しよう

5.5.1 左手法

ソースコード 10 micro_mouse/script/chapter/3/left_hund.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 numpy as np
11 import rospy # include<ros/ros.h>のようなもの
12 from std_msgs.msg import Float32
13 from geometry_msgs.msg import Twist
14
15 import wall
16 import mouse
17
18
19 if __name__ == '__main__': # int main()みたいな
20     rospy.init_node("node_name") # ノード設定
21
22     use_sensor_num = [ 4, 2, 0]
23     sensor = [ 0 for i in range( 3)]
24     sensor_dire = [ 1, 0, -1]
25
26     cell_matrix = [ 17, 17]
27     cell_size = [ 0.500, 0.500]
28     walls = wall.wallPublisher("wall_marker", cell_matrix, cell_size)
29     mouse = mouse.Mouse( "cmd_vel", ["scan/R",
30                                     "scan/FR",
31                                     "scan/F",
32                                     "scan/FL",
33                                     "scan/L"])
34     map_data = np.zeros( cell_matrix)
35
36     vl = vr = 0
37     mx = my = theta = 0
38     try:
39         loop = rospy.Rate(10) # 10[loop/s]の設定をする。
40         while not rospy.is_shutdown():
41             #-----ここにプログラムを書く-----
42
43             #-----センサー読み込み-----
44             for (i, num) in enumerate(use_sensor_num):
45                 sensor[ i] = mouse.sensor[ num].data < 1.0
46             #-----検知した壁を地図に反映させる
47             for (i,dire) in enumerate(sensor_dire):
48                 if sensor[i] == True:
49                     wx = int( round( mx*2+1 + math.cos( (theta + dire) * math.pi/2)))
50                     wy = int( round( my*2+1 + math.sin( (theta + dire) * math.pi/2)))
51                     map_data[ wy, wx] = 1
52             #-----地図情報等を表示する-----
```

```

53     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")

```

5.5.2 拡張手法

ソースコード 11 micro_mouse/script/chapter/3/left_hund_ex.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 numpy as np
11 import rospy # include<ros/ros.h>のようなもの
12 from std_msgs.msg import Float32
13 from geometry_msgs.msg import Twist
14
15 import wall
16 import mouse
17
18
19 if __name__ == '__main__': # int main()みたいな
20     rospy.init_node("node_name") # ノード設定
21
22     use_sensor_num = [ 4, 2, 0]
23     sensor = [ 0 for i in range( 4)]
24     sensor_dire = [ 1, 0,-1,-2] # Left, Forward, Right, Back
25
26     cell_matrix = [ 17, 17]
27     cell_size = [ 0.500, 0.500]
28     walls = wall.wallPublisher("wall_marker", cell_matrix, cell_size)
29     mouse = mouse.Mouse( "cmd_vel", ["scan/R",
30                                     "scan/FR",
31                                     "scan/F",
32                                     "scan/FL",
33                                     "scan/L"])
34     map_data = np.zeros( cell_matrix, dtype=int)
35     map_data_colored = np.zeros( cell_matrix, dtype=int)
36
37     vl = vr = 0
38     mx = my = dire = 0
39     try:
40         loop = rospy.Rate(10) # 10[loop/s]の設定をする。
41         while not rospy.is_shutdown():
42             #-----ここにプログラムを書く-----
43             #-----センサー読み込み-----
44             for (i, num) in enumerate(use_sensor_num):
45                 sensor[ i] = mouse.sensor[ num].data < 1.0
46                 sensor[ 3] = False
47             #-----地図情報を更新する-----
48             #-----壁更新-----
49             tx = mx*2 + 1
50             ty = my*2 + 1
51             for (i,s_dire) in enumerate(sensor_dire):
52                 wx = int( round( tx + math.cos( (dire + s_dire) * math.pi/2)))
53                 wy = int( round( ty + math.sin( (dire + s_dire) * math.pi/2)))
54                 rx = int( round( tx + math.cos( (dire + s_dire) * math.pi/2)*2))
```

```

55     ry = int( round( ty + math.sin( (dire + s_dire) * math.pi/2)*2))
56     if sensor[i] == True:
57         map_data[ wy, wx] = 1 # 壁 ( white)
58         if map_data[ ty, tx] == 0 and map_data[ wy, wx] == 0:
59             if not (rx < 0 or ry < 0 or cell_matrix[0] <= rx or cell_matrix[1] <= ry or i==3):
60                 if map_data[ ry, rx] != 0:
61                     map_data[ wy, wx] = 2 # 仮想的な壁
62                 sensor[i] = (map_data[ wy, wx] != 0)
63     print sensor
64     print mx, my, dire
65     print "-----"
66     #-----道更新-----
67     wall = sum( sensor)
68     if wall== 3:
69         color = 3 # 行き止まり
70     elif wall <= 1:
71         color = 4 # 交差点
72     else:
73         color = 5 # 行き止まりでも交差点でもない場所
74     map_data[ ty, tx] = color
75     #-----マップ色変え & マップ送信-----
76     map_data_colored[ map_data == 0] = walls.WALL_INVISIBLE # 何もない
77     map_data_colored[ map_data == 1] = walls.WALL_WHITE # 壁
78     map_data_colored[ map_data == 2] = walls.WALL_RED # 仮想的な壁
79     map_data_colored[ map_data == 3] = walls.WALL_GREEN # 行き止まり
80     map_data_colored[ map_data == 4] = walls.WALL_ORANGE # 交差点
81     map_data_colored[ map_data == 5] = walls.WALL_GREY # 行き止まりでも交差点でも
82     walls.setData( map_data_colored)
83     #-----行動選択 (左手法)-----
84     vl = vr = 0
85     if sensor[0] == False:
86         vl = 1; vr = 1
87     elif sensor[0] == True and sensor[1] == True:
88         vl = 0; vr = -1
89     elif sensor[0] == True and sensor[1] == False:
90         vl = 1; vr = 0
91     #-----地図やロボット情報を表示する-----
92     walls.publish()
93     #-----ロボットを動かす-----
94     if vr != 0:
95         mouse.move( 0.0, vr * math.pi/2)
96     if vl != 0:
97         mouse.move( vl, 0.0, 10)
98     #-----ロボットの自己位置を更新する-----
99     dire += vr; dire %= 4
100     mx = int( round( mx + vl * math.cos( dire * math.pi/2)))
101     my = int( round( my + vl * math.sin( dire * math.pi/2)))
102     #-----ここまで-----
103     loop.sleep() # 10[loop/s]になるよう調整する。
104     except KeyboardInterrupt: # try:中にCTRL-C が押されればココを実行する。
105         print(" キーボード割り込み ! CTRL-C 終了")

```

5.5.3 足立法

5.6 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