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 8 月 7 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。資料等は github に公開しているので、適時ダウンロードしてください。

```
$ git clone https://github.com/platypus5384/micro_mouse.git
```

また、PC・Ubuntu の基本操作、vim の基本操作、プログラミング言語の基礎知識（if 文・for 文・配列等）が多少 (講義を受けた、1 周間くらい使ったことがある程度) を想定しています。最初は復習から始めますが、日常生活でも練習しておいてください。

2　動作環境について

Ubuntu16.04LTS
ROS Kinetic
での動作を前提としています。Ubuntu18.04LTS 等でも問題ないと思いますが、ros 関連のパッケージ名が違ってくるので注意（kinetic を melodic 等に変えるだけなので簡単）

3 Exercise

3.1 Python **簡易入門**

Python2.7 での動作を想定しています。以下を実行して、2.7.xxx のような出力になれば問題ないです。

```
$ python −V
2.7.12
```

Python を学ぶに至っては「Python 入門」等と調べると、丁寧でわかりやすいサイトが、たくさん多く出てくるので、この PDF ではマイクロマウスシミュレータを触りながら Python を学ぶ事にします。

以下のサイトを見る事を想定して資料を作っています。（以降、このサイトを参考サイトと呼びます。ちなみにこのサイトは別に親切な訳では無いです。）

```
http://tohoho−web.com/python/index.html
```

3.2　マウスシミュレータを起動しよう

マウスシミュレータを起動・試運転させましょう。それぞれ別のタブで実行してください。

```
$ roslaunch micro_mouse start_up.launch
```

実行すると、2 つのウィンドウが開き、以下の図 3.2 のような画面になります。

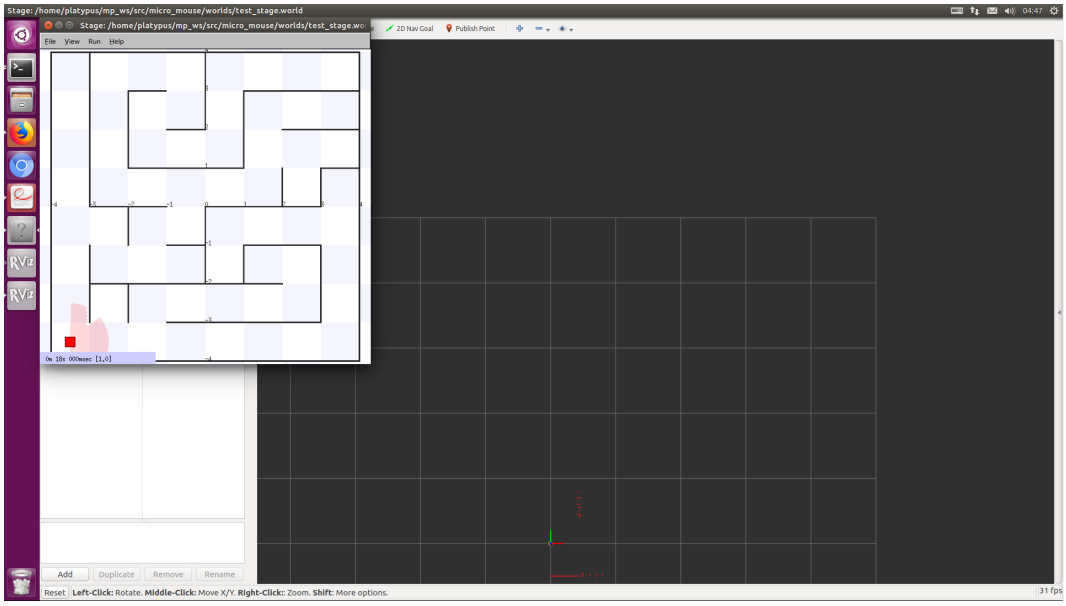

図 1　マイクロマウスシミュレータを実行した時の様子

以下の写真のようなウィンドウはマイクロマウスのシミュレータです。実際のステージと思って貰って大丈夫です。

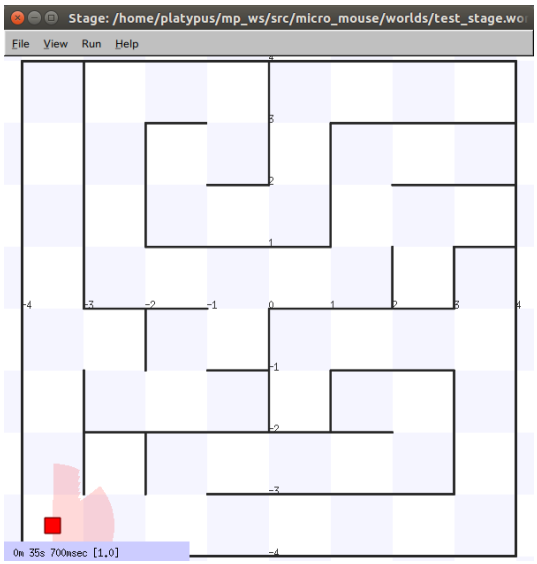

図 2　マイクロマウスシミュレータ

以下の写真のようなウィンドウは Rviz と呼ばれる、情報を可視化するツールです。ROS 標準の物です。上記のシミュレータに直接書き込むことは出来ないので、こちらに壁や地図等を表示させます。実際のロボット開発でも使われており、メカニックカモノハシでも多用します。非常に便利なものなので慣れましょう。

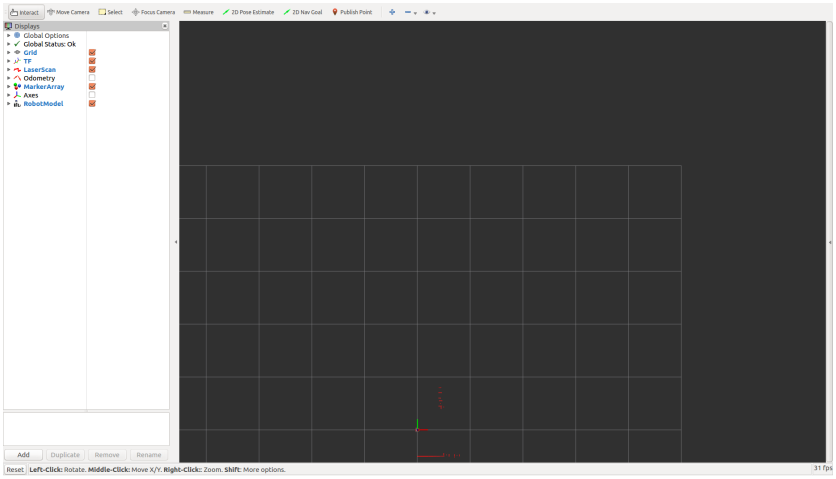

図 3　Rviz

3.3　マウスシミュレータでマウスを動かそう

試運転に以下を実行してみましょう。拡張左手法のプログラムです。

```
$ rosrun micro_mouse left_hund_ex.py
```

実行するとマウスが動き出し、Rviz に壁やタイルが描画されていきます。

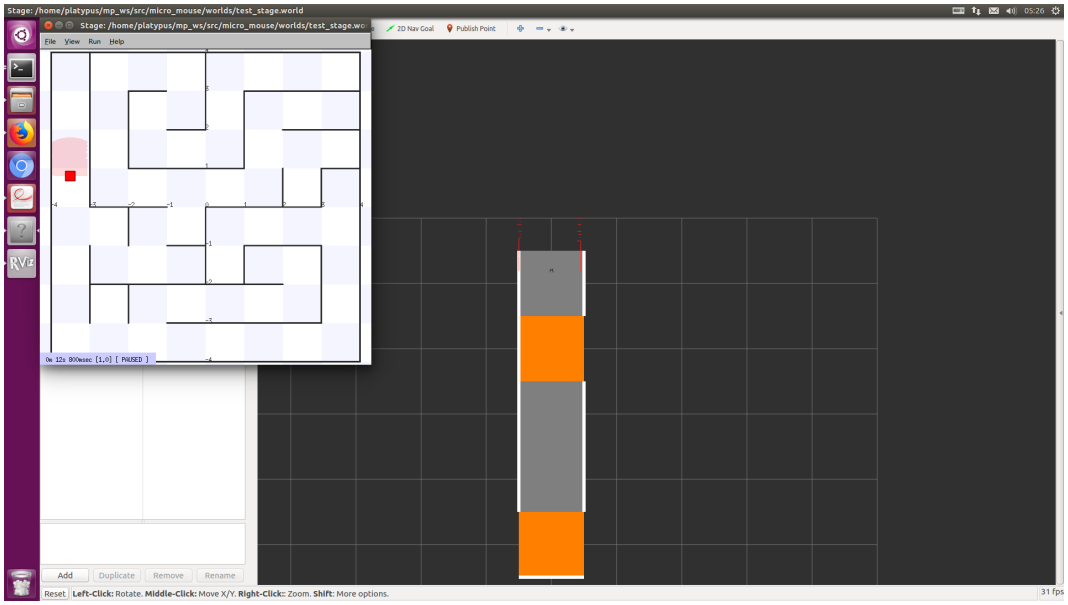

図 4　left_hund_ex.py を実行した直後

せっかくなので、シミュレータの時間停止、加速、減速などを試しましょう。以下を試してみてください。（シミュレータのツールバーに載っています）

```
p 時間停止・開始
[ 減速
] 加速 ( 2 倍速以上?にすると脱線します)
```

3.4 マウスシミュレータをリセットしよう

まずは、プログラム (rosrun) を中断してください。中断したら、以下を実行してください。

```
rosservice call /reset_positions
```

実行するとマウスが初期位置に戻ります。(Rviz は初期化されませんが問題ありません）

3.5 マウスを動かそう１

このセクションでは、以下の知識を前提とします（参考サイト参照)。

[構文]Hello world!(print 文)

3.5.1 サンプルプログラムの解説

1〜25 行目、46〜最終行は、設定のようなものなので無視してください。

print(”¡書きたい文字列¿”) と書くと Terminal 上に表示されます。プログラムのどこが実行されているかを知りたい時に使うと良いでしょう。

マウスを移動・回転させる時には、`mouse.move()` を使います。構文は以下の通りです。

また、速度回転速度制限があり、直進速度は 1[m/s] 、回転速度は π /2 [rad/s] が限界です。ROS は右手系なので、正で半時計回りに回転します。

mouse.move(直進速度 [m/s], 回転速度 [rad/s])

ソースコード 1　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       print("−−−−−−−−−−−−−−−−−−−−")
31       print("マウス動きます。")
32       print("マウス１マス前進します。")
33       mouse.move( 1, 0.0)
34       print("マウス９０度回転します。")
35       mouse.move(0.0, 1 * math.pi/2)
```

```
36          print(”マウス１マス前進します。”)
37          mouse.move( 1, 0.0)
38          print(”マウス１マス後退します。”)
39          mouse.move( −1, 0.0)
40          print(”マウス−９０度回転します。”)
41          mouse.move(0.0, −1 ∗ math.pi/2)
42          print(”マウス１マス後退します。”)
43          mouse.move( −1, 0.0)
44
45  #−−−−−ここまで−−−−−
46          loop.sleep() # 10[loop/s]になるよう調整する。
47      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
48          print(”キーボード割り込み！CTRL−C　終了”)
```

3.5.2 サンプルプログラムの実行結果

実行結果は図 3.5.2 のようになります。

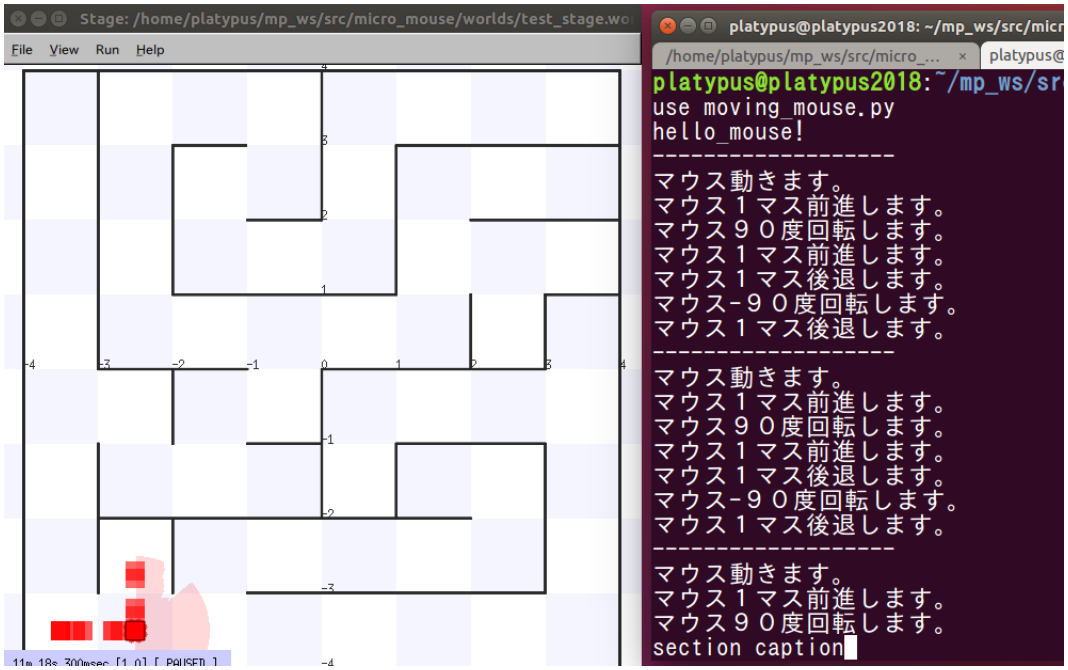

図 5　moving_mouse_result の実行結果

3.5.3 課題

マウスをゴール（右上）まで進ませよう。

3.6 マウスを動かそう 2

このセクションでは、以下の知識を前提とします（参考サイト参照)。

[構文]Hello world! <print 文>
[リスト・タプル・辞書]リスト (list)
[制御構文]もし〜ならば (if, else, elsif)
[制御構文]〜のあいだ (for, in)
[制御構文]ループを抜ける (break)

3.6.1 サンプルプログラムの解説

24〜26 行目に注目してください。ここでは変数を宣言、代入しています。python は変数宣言が必要ありませんが、この PDF では一応明示的に宣言していきます。24 行目ではリスト (配列変数) を初期化しています。25,26 行目は使う変数を初期化しています。

35〜45 行目に注目してください。ここではマウスのプログラムを書いています。

35 行目に注目してください。for 文で action_list の各要素を action に代入しながら、36〜44 行目を繰り返しています。

36〜43 行目に注目してください。if 文で action の数に応じて、vl と vr の値を変えています。

44 行目に注目してください。`mouse.move(vl, vr)` と書く事により、vl,vr による制御を行っています。

つまり、for 文で action_list から要素を 1 つ読み込み、それに応じて動作を行うという事を繰り返しています。

マウスを動かそう 1 では、マウスを指定の位置に移動させる場合、かなりの行数を書かなければなりませんでした。これだと比較的短いコード量・労力で済みんじゃないでしょうか。また、センサの値によって、vl,vr を変えるようなプログラムを作れば自動化も出来そうです。

ソースコード 2　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"])
```

```
#−−−−−変数定義−−−−−
  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
  vl = 0
  vr = 0
#−−−−−変数定義終了−−−−−
  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。
      for action in action_list:
        vl = 0
        vr = 0
        if action == 1:
          vr = math.pi/2
        elif action == 2:
          vl = 1
        elif action == 3:
          vr = −math.pi/2
        mouse.move( vl, vr)
      break
#−−−−−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")
```

3.6.2 サンプルプログラムの実行結果

実行結果は図 3.6.2 のようになります。

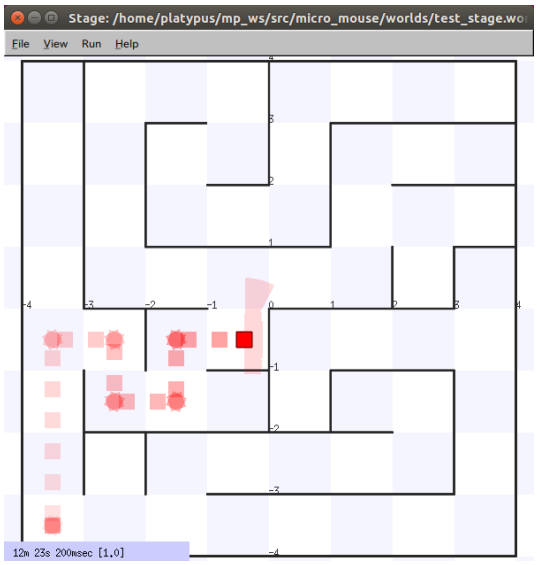

図 6　going_mouse_result の実行結果

3.6.3 課題

マウスをゴール（右上）まで進ませよう。１つ前のセクションよりも、ずっと楽なはずです。

3.7 マウスのセンサを扱おう

このセクションでは、以下の知識を前提とします（参考サイト参照)。

[構文]Hello world! <print 文>
[リスト・タプル・辞書]リスト (list)
[制御構文]もし〜ならば (if, else, elsif)
[制御構文]〜のあいだ (for, in)

3.7.1 サンプルプログラムの解説

26 行目に注目してください。センサのための変数を定義しています。
このように書くと、[0, 0, 0, 0, 0] というような変数を確保できます。37 行目に注目してください。mouse.sensor という配列変数にそれぞれの距離センサ値 [m] が格納されています。右辺式では、センサ値が 1.0[m] 以下かを判定しています 1.0[m] 未満なら True(1)、1.0[m] 以上なら False(0) が帰ってくるので、以降、壁があるなら 1、無いなら 0、と簡単に考えることが出来ます。

42 行目と 45 に注目してください。sensor[2]==0 (壁がない) 時と、else(壁がある無い、ではない時) で処理を書いています。

mouse.sensor[<管理番号>]
管理番号は 18〜22行目を参考にしてください。
L:左、R:右、F:前、FL:左前, FR:右前　です

ソースコード 3　micro_mouse/script/chapter/1/sensing_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", # 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 π /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")
```

3.7.2 サンプルプログラムの実行結果

実行結果は図 3.7.2 のようになります。

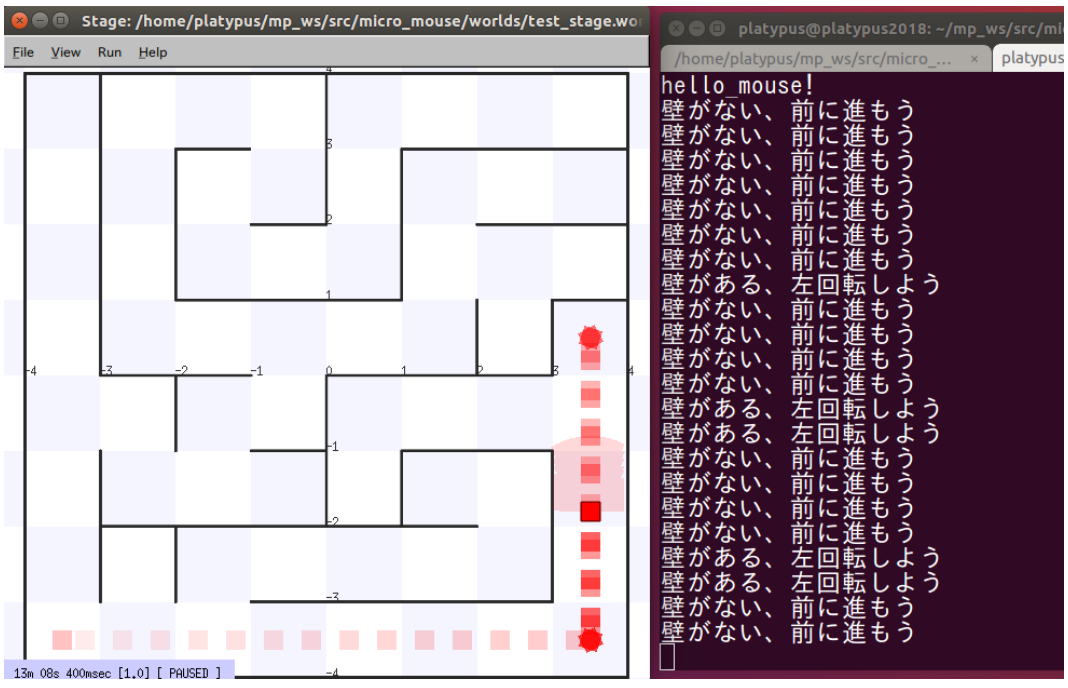

図 7　sensing_mouse_result の実行結果

3.7.3 課題

左に壁がない時、右に壁がない時を検知しよう (print 等で)

3.8 左手法で探索しよう

このセクションでは、以下の知識を前提とします（参考サイト参照)。

[リスト・タプル・辞書]リスト (list)
[制御構文]もし〜ならば (if, else, elsif)
[制御構文]〜のあいだ (for, in)

3.8.1 サンプルプログラムの解説

特にありません。今までのセクションを参考にしてください。

ソースコード 4　micro_mouse/script/chapter/1/left_sensing_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 #−−−−−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   vl = vr = 0
28 #−−−−−変数定義終了−−−−−
29
30   try:
31     loop = rospy.Rate(10) # 10[loop/s]の設定をする。
32     while not rospy.is_shutdown():
33 #−−−−−ここにプログラムを書く−−−−−
34   #−−−−−センサー読み込み−−−−−
35       # mouse.sensor[ 管理番号 ] で、壁までの「距離」が取得できる。
36       # また、( 距離 < 1.0) で、[ 距離 ] を [ ブール値 ] に変える
37       for i in range(5):
38         sensor[ i] = mouse.sensor[ i].data < 1.0
39   #−−−−−行動選択 (左手法)−−−−−
40       vl = vr = 0
41       if sensor[4] == False:
42         vl = 1; vr = 1
43       elif sensor[4] == True and sensor[2] == True:
```

```
44            vl = 0; vr = −1
45          elif sensor[4] == True and sensor[2] == False:
46            vl = 1; vr = 0
47      #−−−−−ロボットを動かす−−−−−
48          if vr != 0:
49            mouse.move(0.0, vr ∗ math.pi/2, 10)
50          if vl != 0:
51            mouse.move( vl, 0.0, 10)
52    #−−−−−ここまで−−−−−
53          loop.sleep() # 10[loop/s]になるよう調整する。
54      except KeyboardInterrupt: # try:中にCTRL−C が押されればココを実行する。
55        print(”キーボード割り込み！CTRL−C　終了”)
```

3.8.2 サンプルプログラムの実行結果

実行結果は図 3.8.2 のようになります。

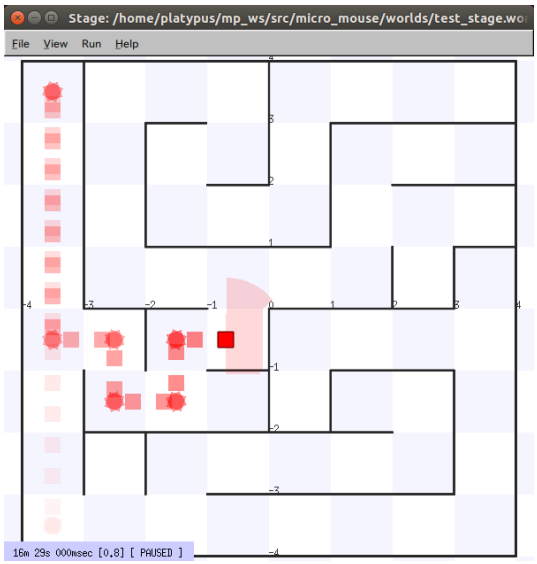

図 8　left_sensing_mouse_result の実行結果

3.8.3 課題

右手法も作ってみよう。また、左・右手法には欠陥があります (ある壁配置だとで全探索が出来ない)。考えよう

3.9 wallPublisher を使いこなそう

3.10 pathPublisher を使いこなそう

3.11 左手法・拡張左手法・足立法を理解しよう

3.11.1 左手法

ソースコード 5　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, dtype="int")
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”)
```

3.11.2 拡張左手法

ソースコード 6　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 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( 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, dtype=int)
  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−−−−−
  #−−−−−センサー読み込み−−−−−
      for (i, num) in enumerate(use_sensor_num):
        sensor[ i] = mouse.sensor[ num].data < 1.0
      sensor[ 3] = False
  #−−−−−地図情報を更新する−−−−−
    #−−−−−壁更新−−−−−
      tx = mx∗2 + 1
      ty = my∗2 + 1
      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))
```

```
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
              ない (踏んだタイル)
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")
```

3.11.3 足立法

3.12 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