チャレンジサイト・メカニックカモノハシ 2019
Ubuntu・Vim の基礎知識
マイクロマウスシミュレータ導入

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 Terminal 及び Vim の基礎知識

Windows は、ほとんどの作業をマウスで操作する事が出来ます (GUI,GraphicalUserInterface)。Ubuntu でもマウス操作をすることが出来ますが、基本的にはコマンド入力による操作 (CUI,CharacterUserInterface　又は CLI,CommandLineInterface) が主になります。慣れてくると、CUI の方が作業効率が良くなる (自由度が高く、script による自動化も簡単に出来る) ので、これを機会に慣れましょう。

このセクションでは、Terminal 及び Vim を使うにあたって作業効率が格段に上がる基礎知識を紹介するので１００%覚えましょう。

3.1 Terminal

デフォルトの Terminal を想定しています。Terminator 等では通用しないかもしれません。

3.1.1 Terminal の開き方

一応言っておくと、Ctrl と Alt はキーボード左下にあります。T は Shift キーを押さずにです。

Ctrl + Alt + T

3.1.2 Terminal 内でのタブの開き方

Ctrl + Shift + T

3.1.3 Terminal 内でのタブ間の移動

Alt + [移動したいタブの左から数えた順番]

3.1.4 Terminal の文字の大きさを変更する

Ctrl + Shift +「+」 Ctrl +「−」

3.1.5 コマンド実行中に強制中断

強制中断出来ない時もあります。

Ctrl + C

3.1.6 Terminal タブを閉じる

タブが一個しかなければウィンドウを閉じます。

Ctrl + D

3.2 Vim

Vim とはエディタです。Vim の強みは CUI 上で動くこと、自由度の高いコマンド操作です。

CUI 上で作業できるのが強みという点は、のちのち理解できると思います。

Vim ではコマンドを打つことで、コピペ、文字列置換やカーソル移動などが行えます。また、Vim のコマンドはプログラムのような物で、自由度が高く使いこなせば作業効率が凄いことになるらしい。プログラムを書く際には正直 Atom や VisualStudio の方が便利ですが、少し書く分には非常に便利で、また、どの PC にも基本的に入っているものなので、是非 Vim に慣れましょう。

以下のサイトにわかりやすく載っていますが、量が多いので最低限これだけは、というのを紹介します。使いながら覚えましょう。

```
https://qiita.com/FBH9999/items/a8ec99bed08592a9d69a
```

3.2.1 Vim のインストール

Vi というのはデフォルトで入っていますが、Vim はデフォルトで入っていせん。ターミナル上で以下を実行してください。($を入力する必要はありません。$から始まる行はターミナル上でという意味です。)

```
$ sudo apt install vim
```

3.2.2 Vim の起動

```
$ vim [ファイル名]
$ vim test.py
$ vim (チュートリアルを開きます)
```

3.2.3 編集モード

編集モード（文章が書ける）に移行します。

```
a (そのカーソル後から編集モードに)
i (そのカーソルから編集モードに)
```

3.2.4 コマンドモード

コマンドモードに移行します。

```
Esc
又は、
Alt + コマンドの一文時目
```

3.2.5 コマンド

```
:w (上書き保存)
:q (終了)
:!q (強制終了 (保存せずに終了))
:wq (保存して終了)
```

:set number 行番号表示
:[数字] その行に移動

以下のコマンドは [数字] + コマンド　とすることで一気にn 回、n 行できます。

h j k l それぞれ､左､下､上､右に移動(矢印でも移動できる)
u (1回作業を戻る (undo))
Ctrl + r (1回作業を進める (redo)
yy (1行コピー)
dd (1行コピー&削除)
p (ペースト)

4 マイクロマウスパッケージの導入

ROS が入っている、環境設定等が終わっている、というのを前提とします。(roscore 出来れば問題なし)

4.1 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。

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

4.2 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。

```
$ 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。

```
$ roscd micro_mouse
```

5　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 1　roscd

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

6　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 2　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 3　roscd

```
$ roslaunch micro_mouse startup.launch
別の端末を開き、
$ rosrun 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。

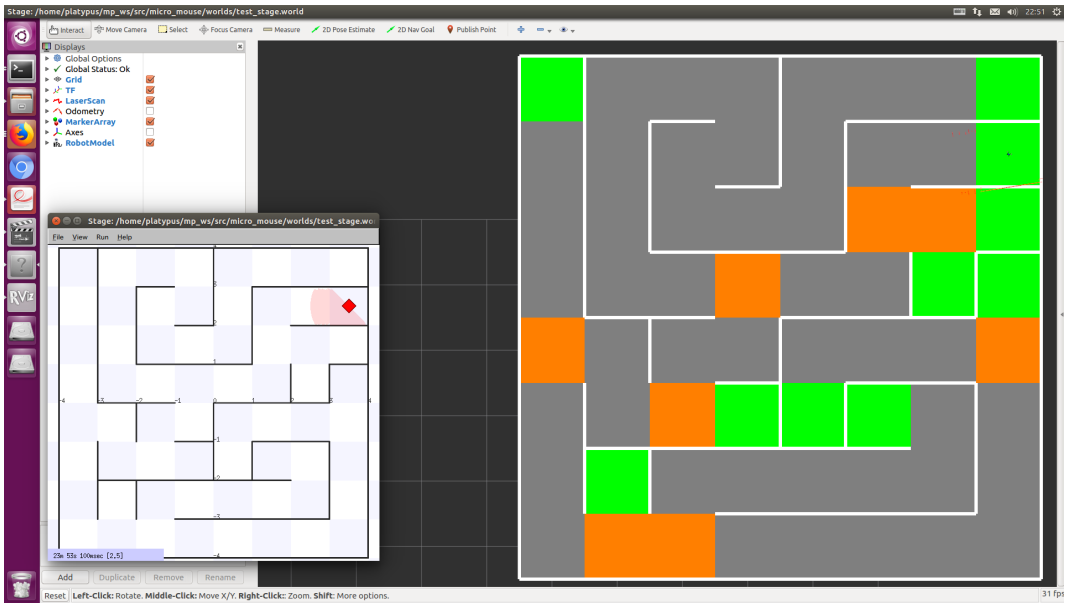

図 1

7 Exercise

7.1 Python **簡易入門**

簡易的に Python 入門しよう。

7.2 マウスを動かそう

ソースコード 4　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 5　micro_mouse/script/chapter/1/going_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"])
#−−−−−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")
```

ソースコード 6　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", # 0 センサの管理番号
                                   "scan/FR", # 1
                                   "scan/F", # 2
                                   "scan/FL", # 3
                                   "scan/L"]) # 4
#−−−−−変数定義−−−−−
  # mouse には 5 つのセンサが搭載されています。
  # 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。
  sensor = [0 for i in range( 5)]
#−−−−−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−−−−−
      #−−−−−センサー読み込み−−−−−
      # mouse.sensor[管理番号]で、壁までの「距離」が取得できる。
      # また、(距離 < 1.0)で、[距離]を [ブール値]に変える
      for i in range(5):
        sensor[ i] = (mouse.sensor[ i].data < 1.0)
  #−−−−−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。

      if sensor[ 2] == 0:
        print("壁がない、前に進もう")
        mouse.move( 1, 0.0)
      else:
        print("壁がある、左回転しよう")
        mouse.move( 0, math.pi/2)

#−−−−−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")
```

7.3 wallPublisher を使いこなそう

7.4 pathPublisher を使いこなそう

7.5 左手法・拡張左手法・足立法を理解しよう

7.5.1 左手法

ソースコード 7　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−−−−−
```

```
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”)
```

7.5.2 拡張左手法

ソースコード 8　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")
```

7.5.3 足立法

7.6 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