

Robot Maze Tracking Report

1. Authors

Group6: SUN YIXUAN 519772745@qq.com
LI YUTONG 1337098103@qq.com
CHEN MENGXUAN 2295266653@qq.com
Address: UM 10th college

2. Abstract

Use the AlphaBet 2 robot to realize the function of walking the maze

3. Introduction

3.1 The principle of robot tracking

The tracking here means that the robot walks along the black line on the white floor. Here, we use the number 1 to represent the black area.

3.2 motor driven

We combine the values (0 or 1) obtained by the five sensors which the program have processed into an array. Depending on the value of the array, perform straight, backward, left, right, turn or stop, and set different parameters (speed, duration, etc.).

3.3 Maze tracking

In maze, the way can generally be divided into eight types: left turn, right turn, T-junction, crossroad, forward or left turn, forward or right turn, dead end, maze end.

The movement of the robot follows the left-hand rule, and the priority of performing the left turn is higher than the straight or right turn; the priority of executing the straight line is higher than the right turn.

4. Related work

The method of our code reference:

①The left hand rule:

<https://www.arduino.cn/thread-22046-1-1.html>

②The right hand rule:

<http://www.61mcu.com/bbs/dispbbs.asp?boardid=6&id=135>

The ideas of the two methods are essentially the same, except that the order of execution is different. It's a good way for beginners like us to understand and get started quickly. Our group's algorithm draws on the left-hand rule.

③Ultrasonic judgment of road conditions:

[HTTPS://blog.csdn.net/hqy2000c/article/details/83044096](https://blog.csdn.net/hqy2000c/article/details/83044096)

<https://blog.csdn.net/hqy2000c/article/details/83188228>

<https://blog.csdn.net/hqy2000c/article/details/83188228>

5. Method

5.1 the main method of the robot

- (1) Go straight along the current track, go straight and check if there is a fork in the front.
- (2) Once the fork is detected, the robot determines the type of fork.
- (3) After determining the type of fork, complete the turn under certain control.

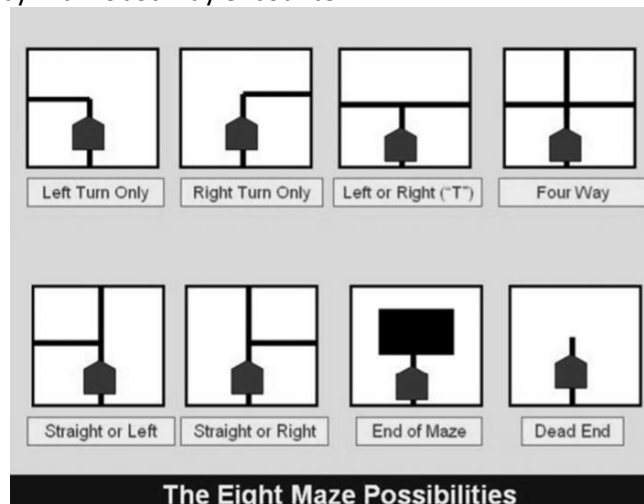
The entire process will cycle as the robot moves through the maze until the robot reaches the end of the maze.

5.2 the robot to find the line method

It is relatively simple for the robot to find the straight line. We only need to adjust the left and right speed of the robot according to the number of sensors. When the robot is directly above the track, the value is $[0,0,1,0,0]$, the robot advances at a speed of 2.5. When the first or second sensor value is 1, it indicates that the car is offset to the right. We should adjust the left and right motor speed to make it fine-tune to the left. When the third or fourth sensor value is 1, it indicates that the car is shifted to the left. We should adjust the left and right motor speed to make it fine-tune to the right. Other situations always progress at a rate of 2.5.

5.3 Identification and treatment of road crossings

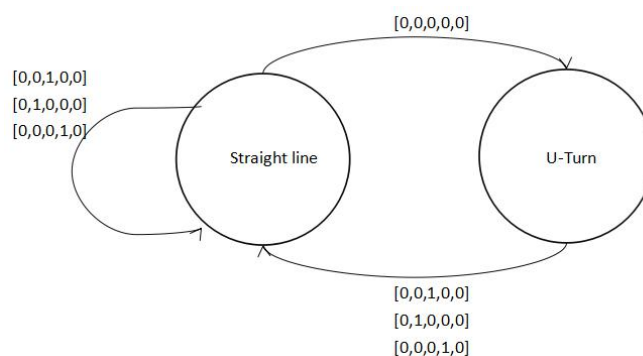
Eight situations that a labyrinth robot may encounter:



(1) Left Turn Only and Right Turn Only

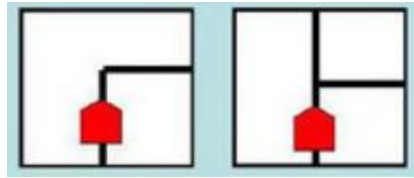
When there are two intersections, Left Turn Only and Right Turn Only, turning left or right is their only choice.

(2) Dead End

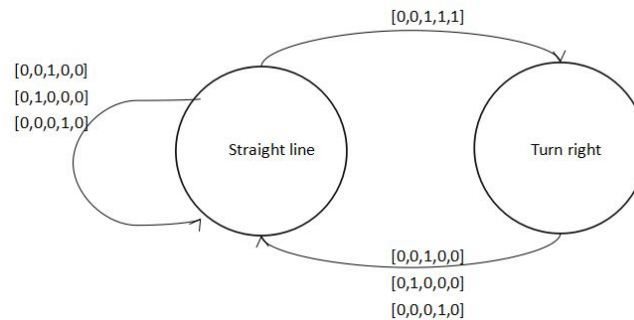


Dead End is the simplest of 8 intersections. When the value of the sensor changes from $[0,0,1,0,0]$ to $[0,0,0,0,0]$, the robot encounters a dead end, and the action it should perform is a U-turn (make a U -Turn).

(3) Right Turn Only and "Straight or Turn right"

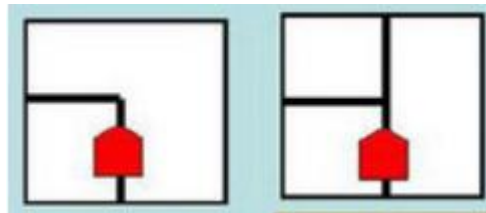


When the robot encounters these two intersections, the sensor's output mode is $[0,0,1,1,1]$.

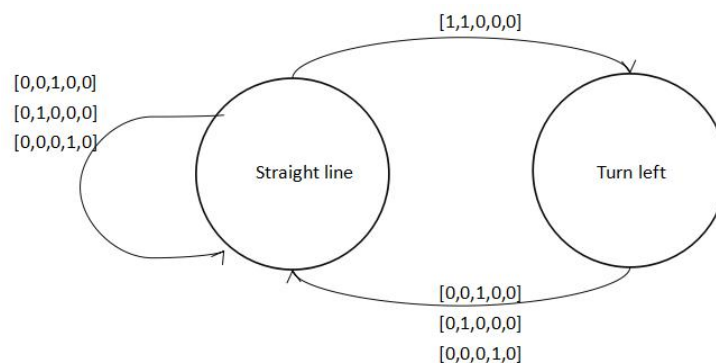


Solution: Write a subroutine named `one()`, and the operation is to let the robot advance a little. When the sensor crosses the track, if the sensor shows $[0,0,0,0,0]$, the intersection is "right turn"; if the sensor shows the value is $[0,0,1,0,0]$ Then, this intersection must be "right turn or straight road", according to the "left hand rule", the robot is required to perform "straight line" action.

(4) Left Turn Only and "Straight or Turn Left"

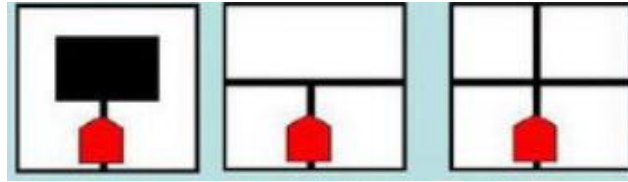


When the robot encounters these two intersections, the output mode of the sensor is $[1, 1, 1, 0, 0]$.



Solution: Write a subroutine named `one()`, and the operation is to let the robot advance a little. When the sensor crosses the track, if the sensor shows $[0,0,0,0,0]$, the intersection is "left turn"; if the sensor shows the value is $[0,0,1,0,0]$ Then, this intersection must be "left turn or straight road". According to the "left hand rule", the robot is required to perform a "left turn" action.

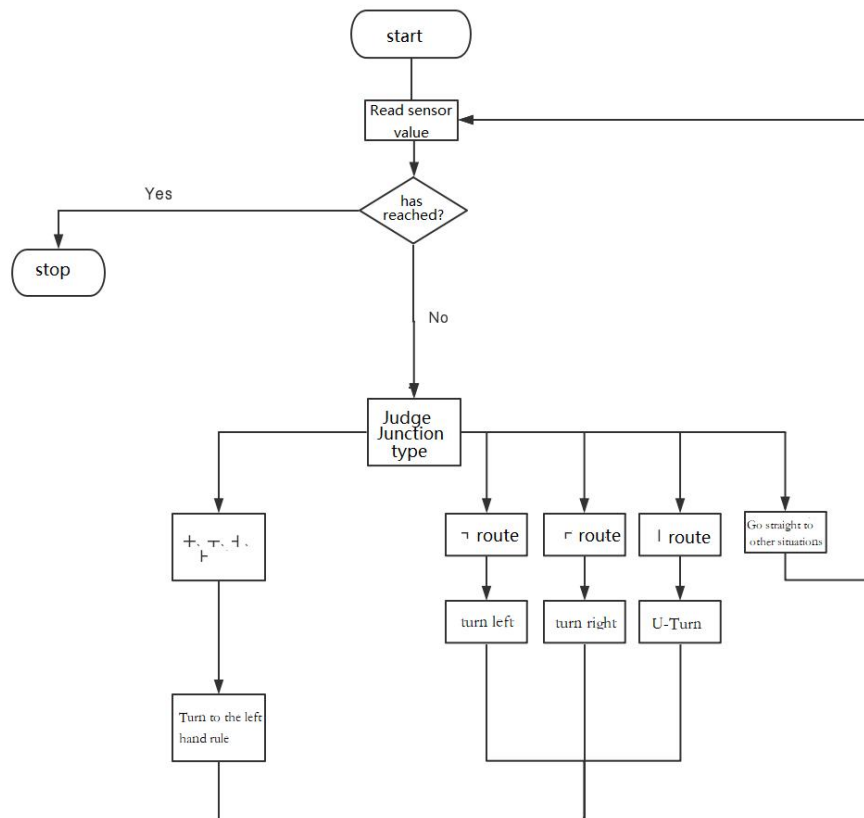
(5) Identification and treatment of "T-junction" , "crossroads" and "endpoints"



When the robot encounters these three intersections, the output mode of the sensor is [1, 1, 1, 1, 1].

Use the one() function to let the robot advance a little further and distinguish three intersections:

If the value of the line-finding sensor is [0,0,0,0,0], then the intersection is a T-junction; If the value of the line-finding sensor is [0,0,1,0,0], then the intersection is an intersection; If the hunt sensor value is [1,1,1,1,1], then this intersection is the end of the maze.



6. Experiments/ Results/ Discussion

6.1 Experiments(About the parameters):

(1) When going straight: the speed of straight walking is 2.5

When the robot needs to make left and right fine adjustments while going straight, the motor speed in the corresponding direction is 1 and the duration is 0.5.

(2) When the robot turns left or right 90° :

If the robot turns to the left, the left wheel speed is 4 and the right wheel speed is 8

The opposite when it turns right

(3) When turning 180 degrees backwards. The speed of the left and right wheels is 8

(4) When the robot tries one step forward: Speed is 1, duration 0.3

(5) Cancel the try: Speed is 1, duration 0.3, backward movement

6.2 Results

The robot is generally operating normally and can handle straight, left turn, right turn, T-junction, dead end.

But for the crossroads and the end of the maze, the robot can't handle it well.

For crossroads: Although it can be detected that it has reached the crossroads, it will not turn to the left first. It will continue to test further before it enters the straight road and begins to go straight.

For the same reason, it will continue to go forward when it has arrive the end of the maze

6.3 discussion

(1) Why do robots keep moving forward?

We let the robot print some extra data while it is running.

In the code, crossroads and T-junctions and destinations are written in a branch (elif).

Indeed, at runtime, the program also entered this branch. However, none of the following sub-branches are executed (and even the else sub-branch is not executed).

I think, on the one hand, is the problem of caching. I also encountered this problem when I was writing the website back end (using the SSH framework). That is, if the changes are particularly small, then re-execute the code ,

the result is likely not to change unless I clear the cache. And this else sub-branch, which I added last (yes we didn't write the else sub-branch at first), so I think it may be that there exist a problem that we didn't clean up the cache, which leads to the case that even the else sub-branch will not run.

On the other hand, the robot constantly tries to move forward, which means that the first three conditions are not satisfied, and there is no else branch, then the parent branch execution ends, thus starting a new round of cycles.

(2)In order to change this phenomenon:

I think that whether it is a T-shaped intersection or a crossroad, it is necessary to turn left first. So they can be written in a sub-branch.

That is to determine whether it is the end point, if it is not the end point, just let the robot turn left just fine.

7. Conclusion/ Future Work

7.1 Difficulties:

To be honest, the whole process is very difficult. On the one hand, we are all just learning the python language. At the beginning we are going to split into two modules. One module is the module that manipulates the robot action, and the other module is used to execute the logic. But there were some problems when calling the action module, so we ended up write them in the one module.


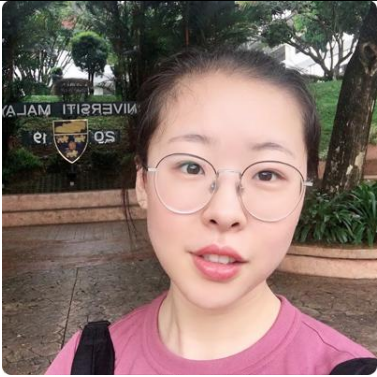
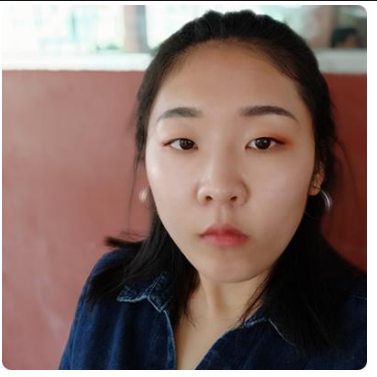
On the other hand, we don't know much about the basics of robotics. At first we don't know how to solve this problem. We have been delayed for a very simple mistake for a long time: we don't know the purpose of 'sleep()', we have learned 'sleep()', but the recognition of it is limited to let the thread sleep (and only learned a very simple application: use 'sleep()' to let multiple threads count in order). Since we didn't write any sleep methods at the beginning, our robots have been unable to get the sensor data, or occasionally get the data, but the walking route is very strange.

Moreover, the movement of the robot has great randomness and uncertainty. If you let it go straight a distance, it is almost impossible to go straight, you need to let the robot know how to adjust by itself. If the adjustment is not good, the error will become larger and larger, which will directly affect the next logic.

7.2 Future work

Let the robot go through the maze correctly.

8. Contributions

		
SUN YIXUAN	LI YUTONG	CHEN MENGXUAN
Responsible for debugging after the code starts running(after linking the robot), writing the rest of the document(part6,7), integrating the documentation.	Responsible for original code writing, debugging, and writing reports in method modules.	Responsible for collecting information; participating in the process of writing and debugging the code (paired programming, she is an observer);responsible for writing the first 4 and ninth modules in the report.

9. References

[HTTPS://www.arduino.cn/thread-22046-1-1.html](https://www.arduino.cn/thread-22046-1-1.html)
<http://www.61mcu.com/bbs/dispbbs.asp?boardid=6&id=135>
[HTTPS://blog.csdn.net/hqy2000c/article/details/83044096](https://blog.csdn.net/hqy2000c/article/details/83044096)
[Https://blog.csdn.net/hqy2000c/article/details/83188228](https://blog.csdn.net/hqy2000c/article/details/83188228)
[Https://blog.csdn.net/hqy2000c/article/details/83188228](https://blog.csdn.net/hqy2000c/article/details/83188228)

10. Codes

```
1. #可以迷宫走的 八种情况
2.
3. from line_tracker import line_tracker
4. from motor import motor
5. from time import sleep
6. address="192.168.0.103"
7.
8. lt=line_tracker(address)
9. sleep(1)
10. mot=motor(address)
11. lt.start()
```

```

12.
13.   thr=370#阈值
14.   route="" # 路径值为空
15.   route_num=0
16.
17.   data_first=[0,0,0,0,0] #初始化数组
18.   done=False #迷宫还未走完全程 如果完成了 为 True
19.
20.   def get_data():#获取值
21.       global data_first
22.       for i in range(5):
23.           if lt.data[i]<thr:
24.               data_first[i]=1
25.           else:
26.               data_first[i]=0
27.       return
28.
29.   def one():# one step 走一步 #在之后调整程序时 可以选择这里 进行微调
30.       mot.command("forward",1,0.3)
31.       stop()
32.       print("one_step")
33.       sleep(1)
34.
35.       return
36.
37.   def one_hou():# one step 走一步 #在之后调整程序时 可以选择这里 进行微调
38.       mot.command("backward",2,0.5)
39.       stop()
40.       print("one_hou")
41.       sleep(1.2)
42.
43.       return
44.
45.   def go_forward():# go forward and Offset adjustment 直走和偏移调整
46.
47.
48.       if data_first == [0,0,1,0,0]:
49.           mot.command("set_left_speed",2.5)
50.           mot.command("set_right_speed",2.5)
51.           mot.command("forward")
52.           print("forward")
53.       elif data_first[3] == 1 or data_first[4] == 1: #如果 3 或 4 寻到黑线 偏左了 应该向右偏移一点
54.           mot.command("right",1,0.5)
55.           print ("litte right")

```

```

56.     elif data_first[0] == 1 or data_first[1] == 1: #如果 1 或 2 寻到黑线 偏右了 应该向左偏移一点
57.         mot.command("left",1,0.5)
58.         print ("litte left")
59.     else:# 此外
60.         mot.command("set_left_speed",2.5)
61.         mot.command("set_right_speed",2.5)
62.         mot.command("forward")
63.         print("forward")
64.         sleep(1)
65.
66. def turn_back(): #turn 180°
67.     while True:
68.         mot.command("set_left_speed",8)
69.         mot.command("set_right_speed",8)
70.         mot.command("left")
71.         get_data()
72.         if data_first[2] == 1 or data_first[1] == 1 or data_first[3] == 1: #如果中间的 检测到黑线 跳出
73.             sleep(0.3)
74.             break
75.         print ("turn_back")
76.         sleep(1)
77.
78. def turn_left():#turn left 90° 左转 直到 传感器 1 2 3 其中一个检测到黑线 跳出
79.     while True:
80.         mot.command("set_left_speed",4)
81.         mot.command("set_right_speed",8)
82.         mot.command("left")
83.         get_data()
84.         if data_first[0] == 1 or data_first[1] == 1 or data_first[2] == 1:
85.             stop()
86.             break
87.         print ("turn_left")
88.         sleep(1)
89.
90.
91. def turn_right():#turn right 90° 右转 直到 传感器 3 4 5 其中一个检测到黑线 跳出
92.     while True:
93.         mot.command("set_left_speed",8)
94.         mot.command("set_right_speed",4)
95.         mot.command("right")
96.         get_data()
97.         if data_first[2] == 1 or data_first[3] == 1 or data_first[4] == 1:
98.             stop()
99.             break

```



```

100.     print ("turn_right")
101.     sleep(1)
102.
103.
104. def stop():#stop
105.     mot.command("set_left_speed",0)
106.     mot.command("set_right_speed",0)
107.     print("stop")
108.     return
109.
110.
111. def routing():#优化路径
112.     route_temp = ""      # 临时变量置空
113.     j = route.length()   # 当前路径的字符串的长度
114.     if j >= 3 and route[j - 2] == 'U':      #如果倒数第二个字符为U（小车掉头了，说明此路不通），路径需要优化
115.         for i in range(j - 3): #将当前路径字符串去掉最后3个字符后赋值给临时字符串，这三个字符将被优化后的1个字符代替
116.             route_temp += route
117.
118.             if route[j - 3] == 'L' and route[j - 1] == 'L': #LUL=S（左转、掉头、左转这3个标志被直行标志S代替，以下类似）
119.                 route_temp += "S"
120.
121.             if route[j - 3] == 'L' and route[j - 1] == 'S': #LUS=R
122.                 route_temp += "R"
123.
124.             if route[j - 3] == 'S' and route[j - 1] == 'L': #SUL=R
125.                 route_temp += "R"
126.             if route[j - 3] == 'R' and route[j - 1] == 'L': #LUL=U
127.                 route_temp += "U"
128.         route.replace(route, route_temp)      #将优化后新的路径赋值给路径字符串 route
129.     return
130.
131. #读取优化后的路径值，选择路口前进方向
132. def cross():#优先级 左 右 直走  读取优化路径中的字符，根据字符执行前进动作
133.     while True:
134.         if route[route_num] == 'L':      #岔路口左转
135.             turn_left()
136.             break
137.         elif route[route_num] == 'R':      #岔路口右转
138.             turn_right()
139.             break
140.         elif route[route_num] == 'S':      #直行穿过岔路口
141.             while True:
142.                 go_forward()
143.                 get_data()

```

```

144.         if data_first[0]==0 and data_first[5]==0:
145.             print("Stright cross")
146.             break
147.
148.     return
149.
150. #主函数
151. while True:
152.     try:
153.         if type(lt.data) == int:
154.             continue
155.
156.         get_data()#取值
157.         print("line_tracker: ",data_first)#打印
158.         #go_forward()#直走
159.
160.
161.         if data_first == [1,1,0,0,0] or data_first == [1,1,1,0,0] or data_first == [1,0,0,0,0]:#┘型路口┐型路口 接下来解
            决方式 前进一小步
162.             print("第一种")
163.             one()
164.             get_data()
165.             turn_left()
166.             #if data_first == [0,0,0,0,0]:#┐型路口
167.                 #turn_left()
168.             #elif data_first[2]==1:#┘型路口 左转或者直行 如果左手定则的话 优先选择左转
169.                 #turn_left()
170.             sleep(0.2)
171.
172.         elif data_first == [0,0,0,1,1] or data_first==[0,0,1,1,1] or data_first == [0,0,0,0,1]:#┐型路 or ┘型路 接下
            来解决方式 前进一小步
173.             print("第二种")
174.             one()
175.             get_data()
176.             if data_first == [0,0,0,0,0]:# 是┐型路
177.                 turn_right()
178.             else:#┐型路 右转或者直行 如果左手定则的话 优先选择直走
179.                 go_forward()
180.                 sleep(0.1)
181.             elif data_first == [0,0,0,0,0]:#死胡同┘
182.                 print("第三种，死胡同
183.                     ")
184.                 one_hou()
185.                 turn_back()

```

```

186.         sleep(1)
187.         elif data_first == [1,1,1,1,1] or data_first == [0,1,1,1,1] or data_first == [1,1,1,1,0] or data_first == [0,
1,1,1,0]:#■终点、或十字型路口或T型路口 三种情况 还是前进一小步判断
188.             print("第四种")
189.             one()
190.             sleep(0.2)
191.             get_data()
192.             sleep(0.1)
193.             print("第四种判断具体形状")
194.             '''if data_first[2] == 0:#T型路口
195.                 print("T型路口")
196.                 turn_left()
197.                 sleep(0.1)
198.             elif data_first[0] == 1 or data_first[1] == 1 or data_first[2] == 1 :#十字型路口
199.                 print("十字型路口")
200.                 turn_left()
201.                 sleep(0.1)
202.             elif data_first == [1,1,1,1,1]:#■终点
203.                 print("■终点")
204.                 stop()
205.                 sleep(0.1)
206.                 done = True #完成循迹
207.                 lt.stop()
208.                 mot.stop()
209.                 break
210.             else:
211.                 turn_right()'''
212.         if data_first==[1,1,1,1,1]:
213.             print("■终点")
214.             stop()
215.             sleep(0.1)
216.             done = True #完成循迹
217.             lt.stop()
218.             mot.stop()
219.             break
220.         else:
221.             turn_left()
222.         else:#其他情况 正常直走
223.             go_forward()
224.
225.         sleep(0.1)
226.
227.     except KeyboardInterrupt:
228.

```

229. `lt.stop()`

230. `mot.stop()`

231. `break`