گزارش تمرین ۳ رباتیک

فربد فولادی - عرفان رفیعی - نگین مشایخی

سوال ۱)

الگوریتم bug 2)

این برنامه شبیه‌سازی کنترل حرکت ربات E-Puck به منظور حرکت از مکان فعلی به مکان مورد نظر (یا هدف) را انجام می‌دهد. در ادامه به توضیح هر بخش از کد خواهیم پرداخت:

1. Import کتابخانه‌ها:

در این بخش، کتابخانه‌های مورد نیاز برای اجرای کد ایمپورت می‌شوند. این کتابخانه‌ها شامل `sys` برای مدیریت سیستم، `math` برای انجام عملیات ریاضی، `random` برای تولید اعداد تصادفی و `controller` برای اتصال به شبیه‌ساز ویباتس هستند.

2. مسیر وباتس:

در این قسمت مسیر ماژول‌های کتابخانه وباتس برای ایمپورت در کد تعیین می‌شود و به `sys.path` اضافه می‌شود.

3. مقداردهی اولیه:

ابتدا شی ربات E-Puck ایجاد می‌شود و سپس موتورهای چرخ چپ و راست به طور بی‌نهایت تنظیم می‌شوند. همچنین، سنسورهای GPS و قطب‌نما فعال می‌شوند.

4. تنظیمات اولیه:

در این بخش تعدادی تنظیمات اولیه مانند دوره نمونه‌برداری، مقادیر اولیه مکان و جهت ربات، و متغیرهای مربوط به حرکت ربات تعریف می‌شوند.

5. تعریف توابع:

در این بخش تعدادی تابع برای محاسبه جهت ربات با استفاده از قطب‌نما، بررسی تطابق با خط عمودی و افقی، محاسبه فاصله اقلیدسی بین مکان فعلی و هدف، و بررسی وجود موانع در جهت حرکت ربات تعریف شده‌اند.

6. تابع بررسی وجود مانع در جهت حرکت:

این تابع با دریافت زاویه جهت ربات، لیستی از زوایا و شاخص‌های سنسورهای سونار فراهم شده و بررسی می‌کند که آیا موانعی در جهت حرکت ربات وجود دارند یا خیر.

7. تابع has_clear_path:

این تابع با دریافت زاویه جهت ربات، موقعیت عمودی فعلی، و موقعیت هدف، بررسی می‌کند که آیا مسیری بدون مانع برای حرکت ربات در جهت مورد نظر وجود دارد یا خیر.

8. تابع محاسبه سرعت چرخش:

این تابع با دریافت جهت فعلی ربات و زاویه هدف، سرعت‌های چرخش چرخ‌های چپ و راست را محاسبه می‌کند تا ربات به سمت جهت هدف چرخش کند.

9. تابع navigate_towards_goal:

این تابع با دریافت موقعیت فعلی ربات و موقعیت هدف، به همراه جهت فعلی ربات، تصمیم می‌گیرد که ربات باید به سمت هدف حرکت کند یا چرخش کند.

10. توابع `navigate_following_right_wall` و `navigate_following_left_wall`:
این دو تابع به منظور حرکت ربات در امتداد دیوارهای راست و چپ تعریف شده‌اند. آن‌ها بر اساس وجود دیوارها و موقعیت ربات تصمیم می‌گیرند که ربات باید چگونه حرکت کند.

11. بخش اصلی:
در این بخش از یک حلقه تکرار برای انجام محاسبات و حرکت ربات به سمت هدف استفاده می‌شود. موقعیت و جهت فعلی ربات به‌روزرسانی شده و وجود دیوارها و موانع بررسی می‌شود. ربات به سمت هدف حرکت می‌کند تا زمانی که به هدف برسد یا موانعی در جلویش وجود داشته باشد. اگر مسیر روشن شود، ربات به سمت هدف حرکت خواهد کرد و در غیر این صورت به سمت دیوارها حرکت می‌کند. همچنین، در صورت رسیدن به هدف، حرکت متوقف می‌شود.

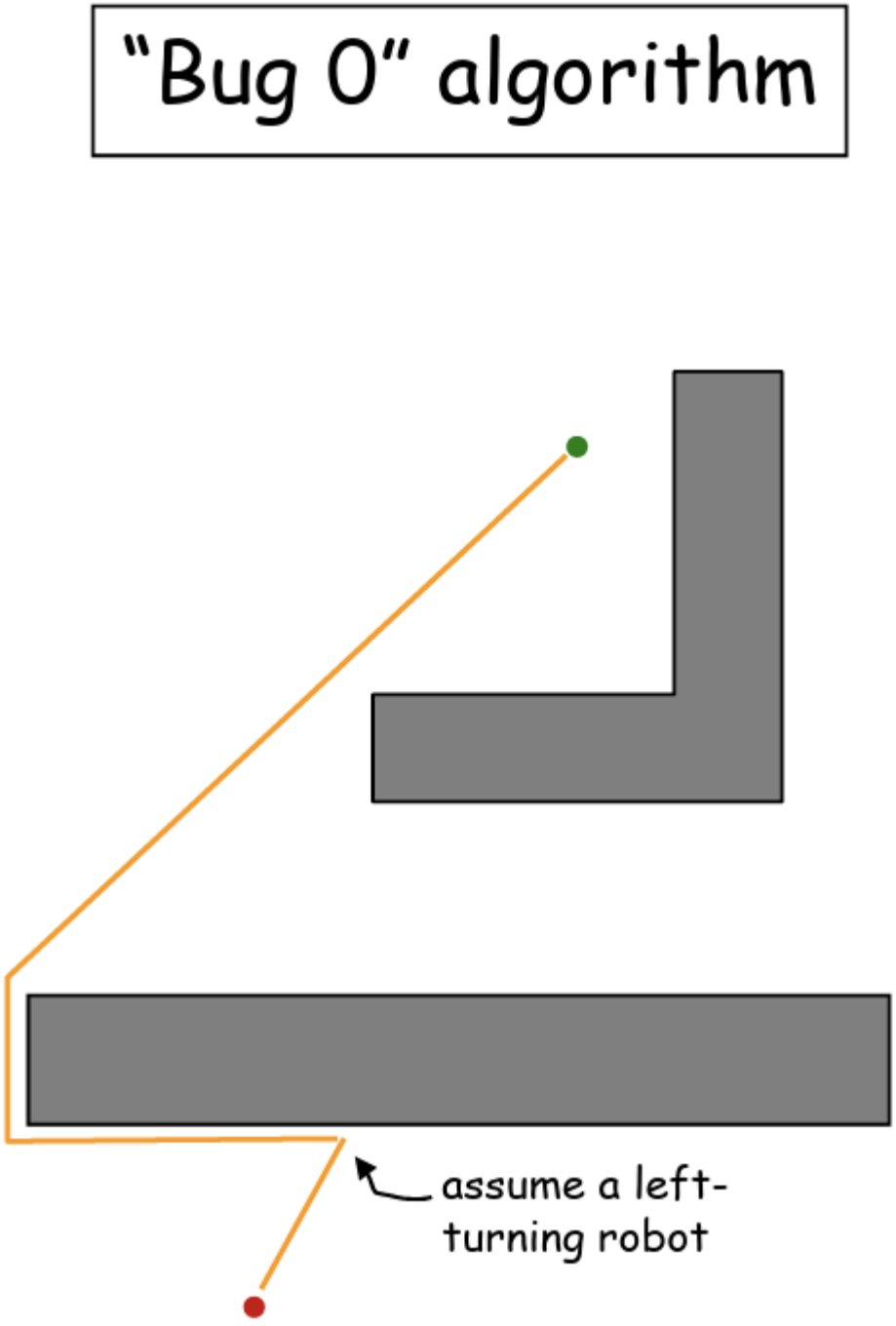
الگوریتم bug1 :

در این بخش ۳ استیت داریم:
استیت ۰: زمانی که جلوی ربات دیواری وجود ندارد: در این صورت ربات یک خط مستقیم فرضی از موقعیت خود تا مقصد رسم میکند و آنقدر میچرخد تا جهت سر ربات در راستای خط فرضی باشد و سپس روی خط مستقیم حرکت میکند تا به دیوار یا هدف برسد.
اگر به هدف برسد تمام میشود، اگر به دیوار برسد وارد استیت ۱ میشود.
استیت ۱: زمانی که در مسیر ربات یک مانع وجود دارد: اینجا الگوریتم آنقدر دیوار را تعقیب میکند تا یک دور کامل دور دیوار بچرخد و در همین حین در هر استپ فاصله خود با هدف را محاسبه میکند. و مینیمم فاصله را با لوکیشن مربوطه به خاطر میسپارد.
وقتی یک دور کامل چرخید دور هدف به استیت ۲ میرود.
استیت ۲: در اینجا ربات کامل مانع را دیده و میداند نزدیک ترین نقطه جدا شدن از دیوار کجاست و دوباره دیوار را تعقیب میکند تا به آن نقطه برسد. زمانی که رسید وارد استیت ۰ میشود و دوباره روی خط مستقیم ادامه میدهد تا به هدف یا دیوار بعدی برسد.

الگوریتم باگ 0 :

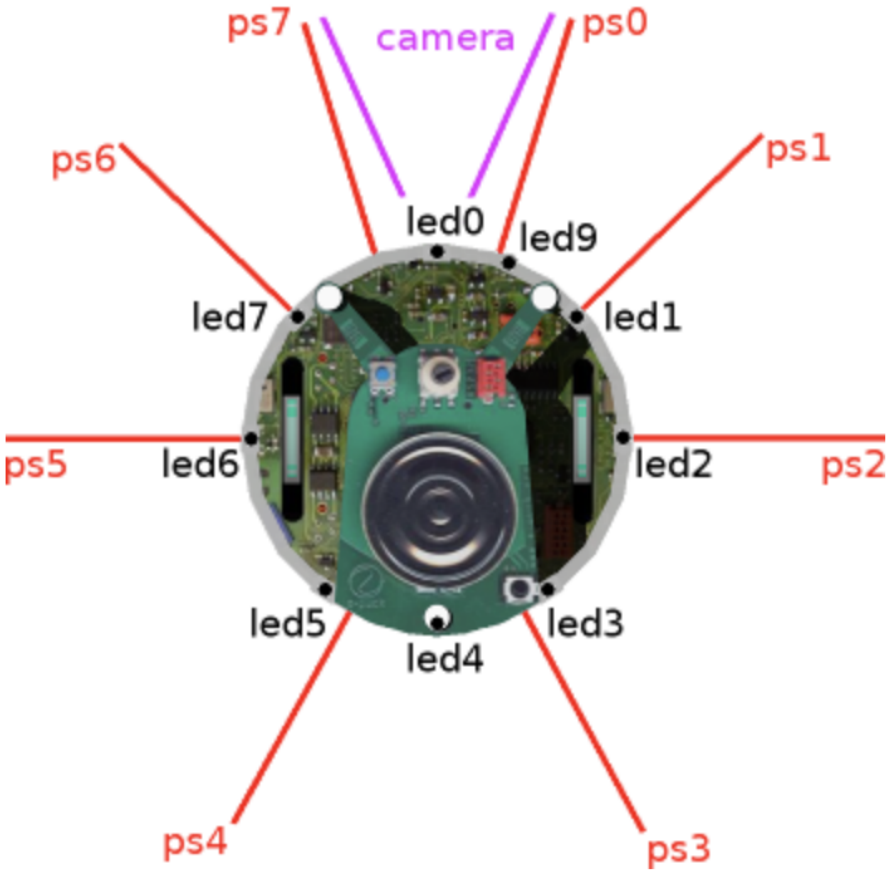
در این الگوریتم با ایجاد تغییراتی در کد باگ 2 کاری کردیم که طبق تعریف ربات به سمت هدف حرکت کرده و در مواقع مواجهه با دیوار آنرا دور زده و در اولین فرصت جدایی از دیوار جدا شود و به سمت هدف حرکت کند.

طبق عکس زیر :



سوال ۲)

ربات e-puck مجهز به ۸ سنسور فاصله در مختصات زیر است.



Sensors, LEDs and camera

ما برای تعقیب

دیوار از سنسورهای سمت چپ یعنی ps5,ps6,ps7 استفاده کردیم.

از طرفی برای بدست آوردن موقعیت ربات از gps استفاده کردیم.

```
("gps_sensor = robot.getDevice("gps
(gps_sensor.enable(1
('sonar5 = robot.getDevice('ps5
(sonar5.enable(1
('sonar6 = robot.getDevice('ps6
(sonar6.enable(1
('sonar7 = robot.getDevice('ps7
(sonar7.enable(1
```

لاجیک اصلی کد به این صورت است:

```
:if not first_move
left_speed,right_speed = max_speed,max_speed
```

در ابتدا ربات برای پیدا کردن اولین دیوار باید انقدر مستقیم برود که به یک دیوار برسد.نقشه ماز هرطور باشد باید باید اول ربات را به یک نقطه شروع رساند.

```
:if sonar7.getValue() > 80
first_move =True
```

اولین باری که ربات یک دیوار را ببیند وارد بخش اصلی الگوریتم یعنی تعقیب دیوار میشود

```
:else
:if sonar7.getValue() > 80
('print('Front wall #
left_speed = max_speed
right_speed = -max_speed
```

این حالت برای زمانیست که ربات با یک دیوار مقابلش مواجه میشود و باید به راست بچرخد که دیوار سمت چپش باشد(ما همیشه در این الگوریتم دیوار سمت چپ را دنبال میکنیم)

```
:elif sonar6.getValue() > 80
('print('Left corner #
left_speed = max_speed
```

```
right_speed = -max_speed
```

زمانی که ربات به یک گوشه چپ برسد باید درجا به سمت راست بچرخد

```
:elif sonar5.getValue() > 80
('print('Wall following #
left_speed = max_speed
right_speed = max_speed
```

اگر ربات در سمت چپ خود دیوار ببیند مستقیم میرود

```
:else
('print('Right corner #
left_speed = 0
right_speed = max_speed
```

اینجا زمانی است که به گوشه راست میرسیم و باید حول مرکز(نقطه ی گوشه) ۹۰ درجه به چپ بچرخیم.

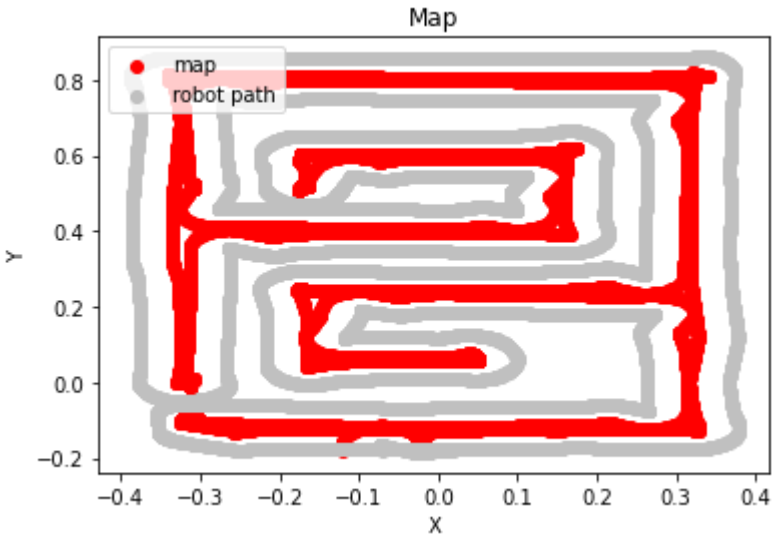
این الگوریتم تا زمانی ادامه دارد که به نقطه شروع میرسیم.

در هر استپ زمانی با استفاده از gps مختصات ربات و با استفاده از compass زاویه ی سر ربات را در فایل ذخیره کردیم.کد دیگری با عنوان q2_maping وجود دارد که این اطلاعات را از فایل میخواند و با دانستن مکان ربات، جهت ربات و فاصله ی عرضی ربات با دیوار مختصات دیوار را محاسبه میکند و نقشه را میکشد.

فومول محاسبه مختصات دیوار به شکل زیر است:

```
x_wall = x_robot + sin_teta * distance
y_wall = y_robot - cos_teta * distance
```

خروجی آن نیز به شکل زیر است.



سوال ۳)

در سوال قبل الگوریتم تعقیب دیوار را پیاده سازی کردیم با این تفاوت که موقع رسیدن به هدف ادامه ربات مسیر خود را تا خارج شدن از ماز ادامه میداد.در اینجا با اضافه کردن یک شرط که با محاسبه فاصله تا هدف رسیدن به هدف را چک میکند الگوریتم را خاتمه می دهیم.

```
if not first_move: # first move
left_speed,right_speed = max_speed,max_speed
:if sonar7.getValue() > 80
first_move =True
elif math.sqrt((0-location[0])*(0-location[0]) + (-0.5-location[1])*(-0.5-location[1])) <
0.05: # goal
('print('End
break
```

سوال 4)

طبق جدول داده شده، و اطلاعاتی که داریم. حرکت ربات به صورت زیر است :

$\Delta x=1.0$ and $\Delta y=(0.0 | 1.0)$

پس یعنی حرکت ما یا به صورت یک خانه به راست است یا یک خانه به سمت راست و بالا.

طبق جدولی که برای bel(x0) داریم که به صورت زیر است :

0.02	0.01	0.02	0.01	0.01
0.01	0.04	0.02	0.01	0.00
0.03	0.40	0.20	0.01	0.03
0.02	0.01	0.05	0.04	0.01
0.00	0.01	0.01	0.01	0.02

پس طبق چیزی که گفتیم، دو حالت داریم که میتوان به خانه 3,3 رسید. یکی از خانه 2,2 و یکی از خانه 2,3. حال با روش بیز احتمال را حساب میکنیم :

$$\overline{bel}(x_1=3, y_1=3) = P(x_0=2, y_0=3) \times P(u_x=1, u_y=0)$$
$$+ P(x_0=2, y_0=2) \times P(u_x=1, u_y=1) =$$
$$0.4 \times 0.5 + 0.01 \times 0.5 = 0.205$$

سپس با مقداری که از η.p(z1|x1,M) = 2 داریم جواب نهایی را نرمالایز میکنیم :