دانشگاه صنعتی اصفهان

دانشکده برق و کامپیوتر

عنوان موضوع و زمینه اصلی:

ارائه کتبی تمرین دوم رباتیک

استاد درس:

دکتر پالهنگ

دانشجو:

داریوش حسن پور

هوش مصنوعی (9308164)

1393

1 - قسمت اول تمرین

در این قسمت بایستی تلاش شود که با داشتن مشخصات فیزیکی ربات شامل شعاع چرخ های و نصف فاصله‌ی بین دو چرخ یک ربات دیفرانسیلی و همچنین با داشتن سرعت زاویه‌ای چرخ ها بتوان مسیری را که ربات طی خواهد کرد را نشان داد.

معادلات (1) و (2) و (3) سرعت ربات را بر بروی هر یک از محور های x و y و همچنین سرعت زاویه‌ای ربات را با داشتن سرعت های زاویه‌ای چرخ های ربات و شعاع و نصف فاصله‌ی بین چرخ‌ها را می‌دهند.

$$\dot{x} = \frac{r}{2}(\dot{\varphi}_r + \dot{\varphi}_l)\cos(\theta) \tag{1}$$

$$\dot{y} = \frac{r}{2}(\dot{\varphi}_r + \dot{\varphi}_l)\sin(\theta) \tag{2}$$

$$\dot{\theta} = \frac{r}{2l}(\dot{\varphi}_r - \dot{\varphi}_l) \tag{3}$$

برای بدست آوردن مکان و جهت ربات از معادلات بالا ابتدا لازم است از سرعت زاویه‌ای ربات انتگرال گرفته شود و سپس با داشتن زاویه‌ی ربات با انتگرالگیری از سرعت‌های مکانی ربات مکان ربات را بدست آورد.

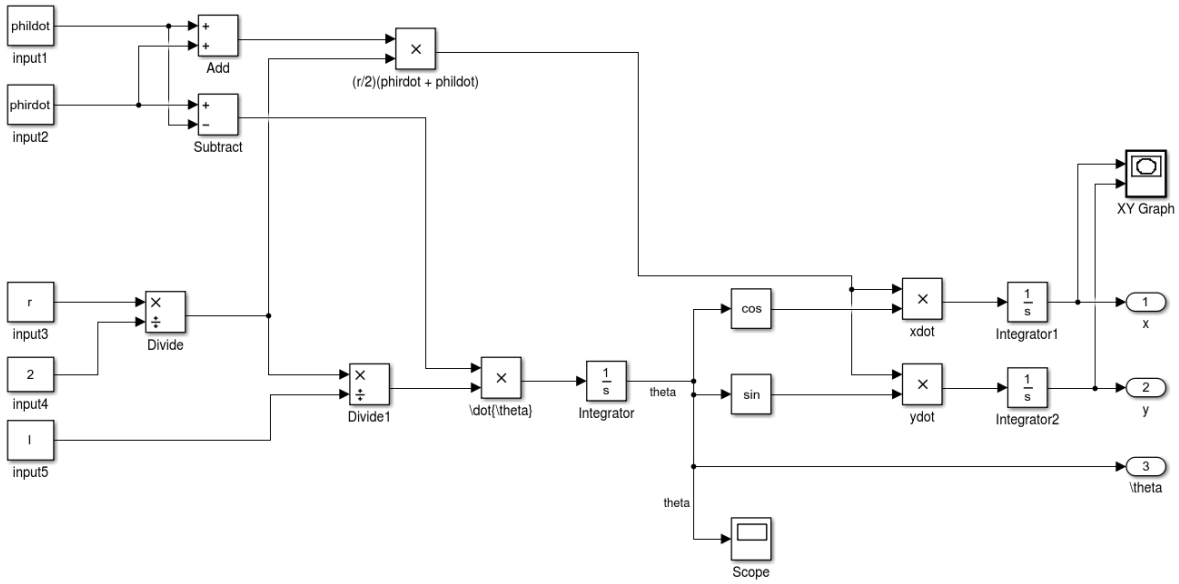
$$x = \int \frac{r}{2}(\dot{\varphi}_r + \dot{\varphi}_l)\cos(\theta)dt = \frac{r}{2}(\dot{\varphi}_r + \dot{\varphi}_l)\cos(\theta)t + x_0 \tag{4}$$

$$y = \int \frac{r}{2}(\dot{\varphi}_r + \dot{\varphi}_l)\sin(\theta)dt = \frac{r}{2}(\dot{\varphi}_r + \dot{\varphi}_l)\sin(\theta)t + y_0 \tag{5}$$

$$\theta = \int \frac{r}{2l}(\dot{\varphi}_r - \dot{\varphi}_l)dt = \frac{r}{2l}(\dot{\varphi}_r - \dot{\varphi}_l)t + \theta_0 \tag{6}$$

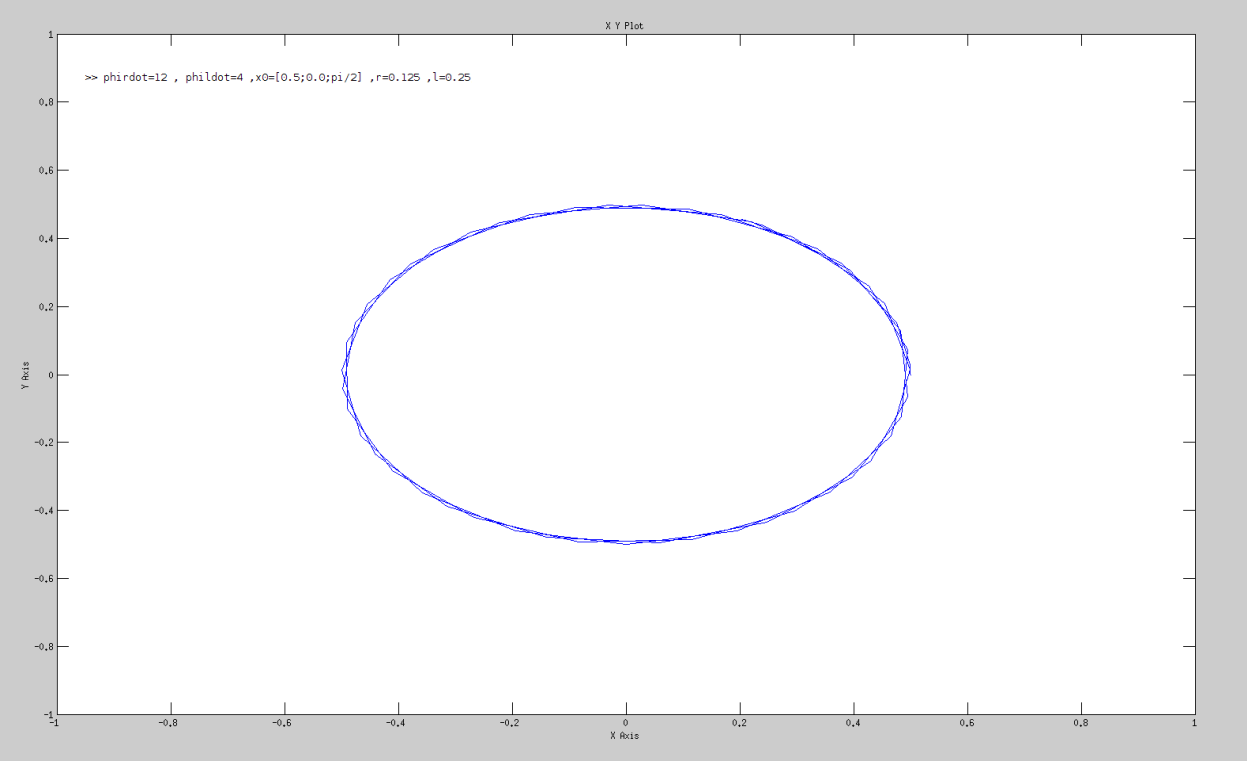
حال کافیست که معادلات (4) و (5) و (6) را در سیمولینک[1] پیاده سازی کنیم که به صورت شمای شکل شماره 1-1 خواهد شد.

[1] Simulink

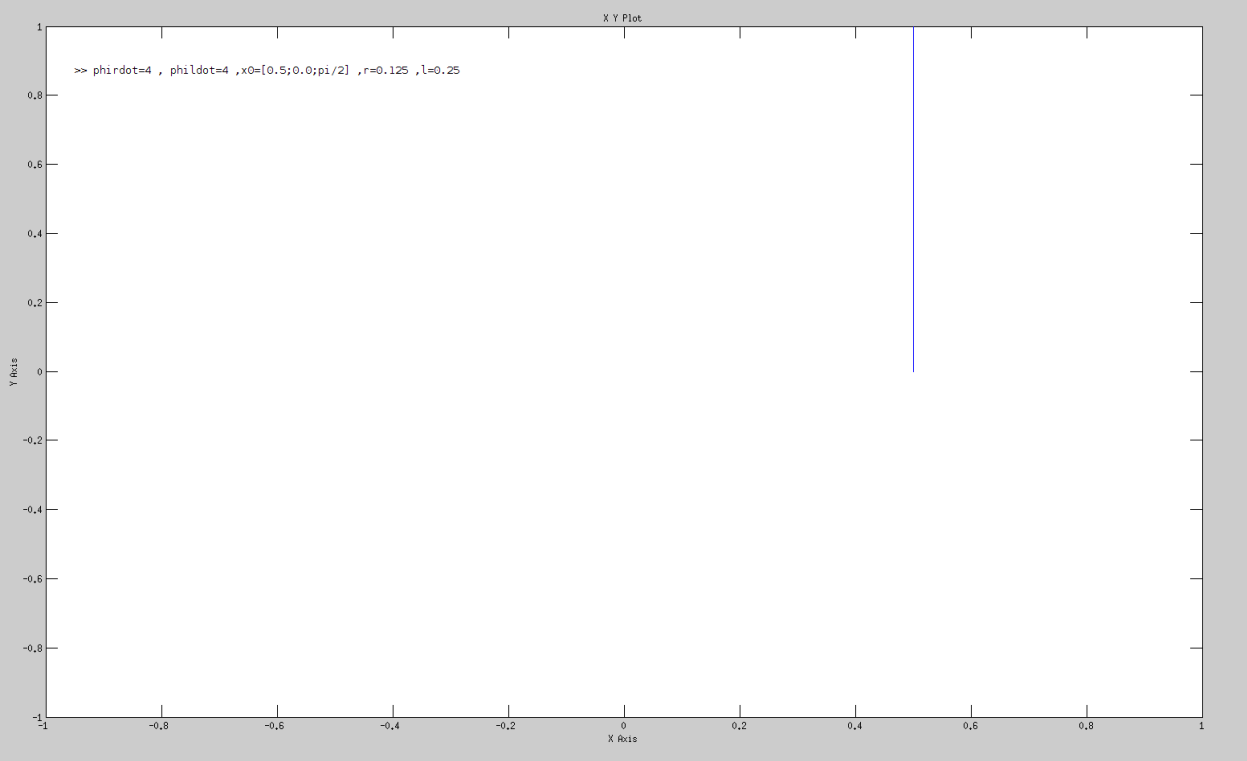


شکل 1-1: پیاده سازی معادلات (4) و (5) و (6)

که چند عدد تست جهت اطمینان از کار کرد مدار معادلاتی شکل 1-1 انجام شده که در شکل های 1-2 و 1-3 آمده است که مدار شکل 1-1 نتایج مورد انتظار را فراهم کرده است.



شکل 1-2: نحوه‌ی عملکرد مدار شکل 1-1 با مقادیر مثالی موجود در متن تعریف تمرین



شکل 1-3: با داشتن سرعت های زاویه‌ای برابر باید ربات در یک مسیر مستقیم حرکت کند.

2 - قسمت دوم تمرین

در این قسمت هدف پیاده‌سازی یک کنترل کننده برای ربات دیفرانسیلی میباشد که بتواند با داشتن موقیت‌های شروع و پایان؛ ربات را در مسیری به حرکت وا دارد که در نهایت به موقیت نهایی اعلام شده برسد.

موقعیت‌ها باید دارای ویژگی‌های

1. مختصات دکارتی ربات
2. زاویه ربات در مختصات جهانی

باشد.

$$position = \begin{bmatrix} x & y & \theta \end{bmatrix}$$

شکل 2-1: نحوه ی نمایش موقعیت ربات

که موقعیت ربات در موقعیت اولیه و نهایی به ترتیب توسط نماد های x0 و xg نمایش داده میشوند:

$$x0 = \begin{bmatrix} x_0 & y_0 & \theta_0 \end{bmatrix}$$
$$xg = \begin{bmatrix} x_g & y_g & \theta_g \end{bmatrix}$$

که با داشتن دستور های کنترلی معادلات (7) و (8) باید بتوانیم که ربات را به موقعیت دلخواه هدایت کنیم.

$$\upsilon = \kappa_\rho \rho \tag{7}$$
$$\omega = \kappa_\alpha \alpha + \kappa_\beta \beta \tag{8}$$

که دارای ثابت‌های زیر هستند:

$$\begin{bmatrix} \kappa_\rho \\ \kappa_\alpha \\ \kappa_\beta \end{bmatrix} = \begin{bmatrix} 3 \\ 8 \\ -1.5 \end{bmatrix}$$

در معادلات (7) و (8) باید به محاسبات روابط (9) و (10) و (11) بپردازیم:

$$\rho = \sqrt{\Delta x^2 + \Delta y^2} \tag{9}$$

$$\alpha = \arctan 2(\frac{\Delta y}{\Delta x}) - \theta \tag{10}$$

$$\beta = \theta_g - \alpha - \theta = \theta_g - \arctan 2(\frac{\Delta y}{\Delta x}) \tag{11}$$

در روابط (9)تا (11) تعریف (12) را داریم:

$$\Delta\Psi \equiv \Psi_2 - \Psi_1 \tag{12}$$

براي يک ربات ديفرانسيلي، مدل سينماتيکي بوسيله‌ی معادلات (13) و (14) بيان مي شود:

$$\upsilon = \frac{r}{2}(\dot{\varphi}_r + \dot{\varphi}_r) \tag{13}$$

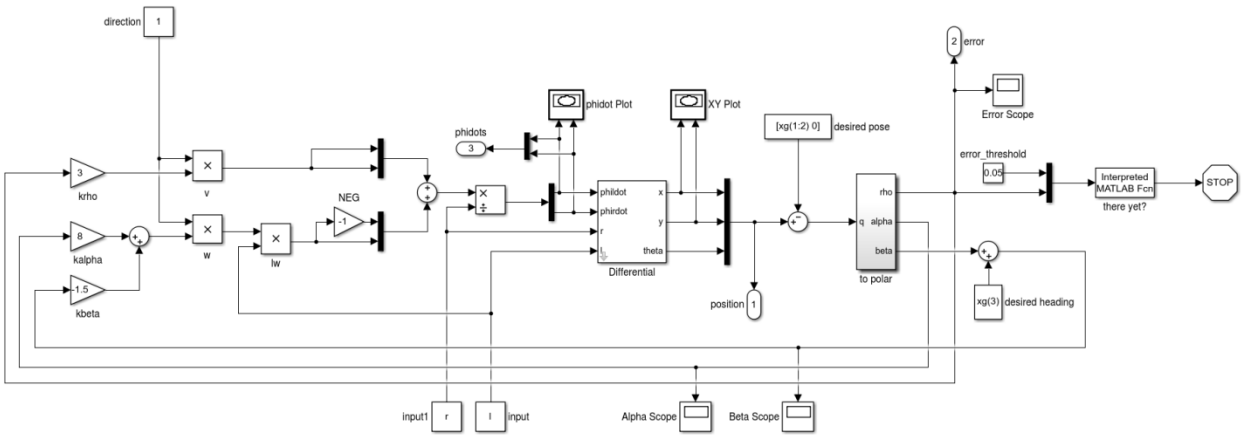
$$\omega = \frac{r}{2l}(\dot{\varphi}_r - \dot{\varphi}_r) \tag{14}$$

حال برای کنترل ربات ابتدا رابطه‌اي بدست آوريم که بتوان سرعت چرخهاي ربات را از روي سرعت خطي و سرعت زاويه‌اي ربات بدست آورد که به معادلات (15) و (16) ميرسيم:

$$\dot{\varphi}_l = \frac{\vartheta - l\omega}{r} \tag{15}$$

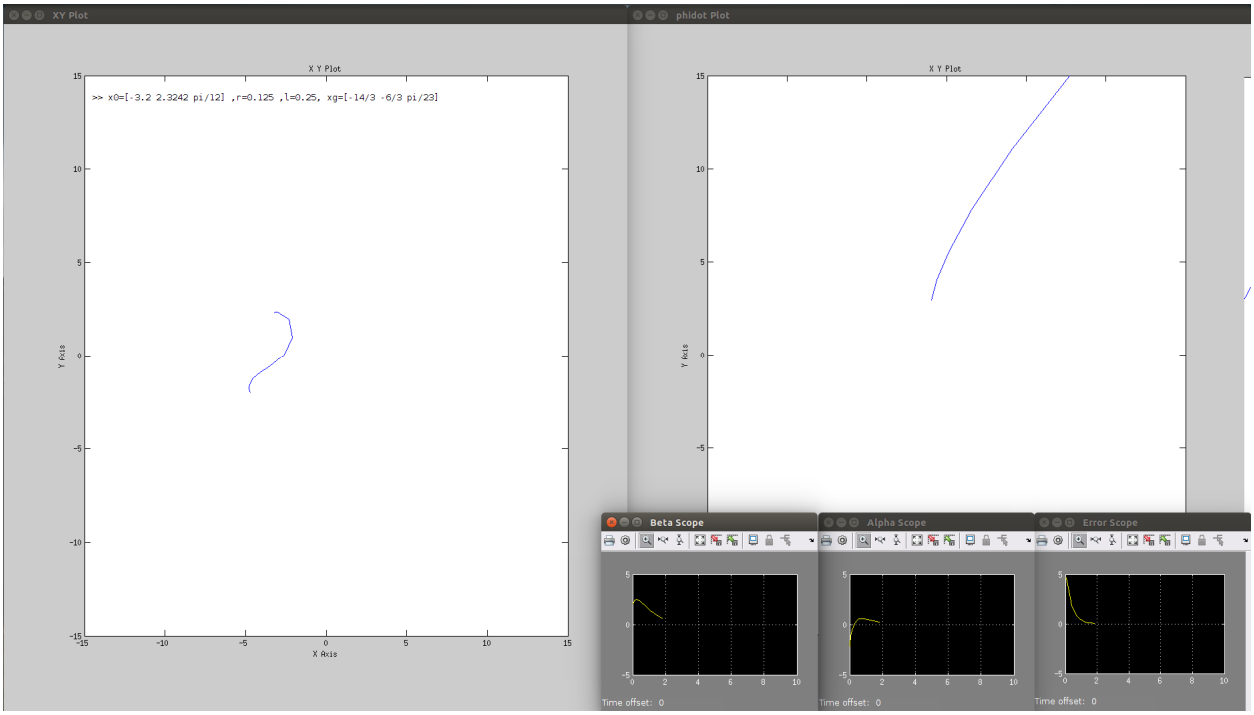
$$\dot{\varphi}_r = \frac{\vartheta + l\omega}{r} \tag{16}$$

با استفاده از مدار شکل 1-1 در و روابط (7)تا (16) مدار جهت کنترل ربات به صورت شکل 2-2 خواهد بود

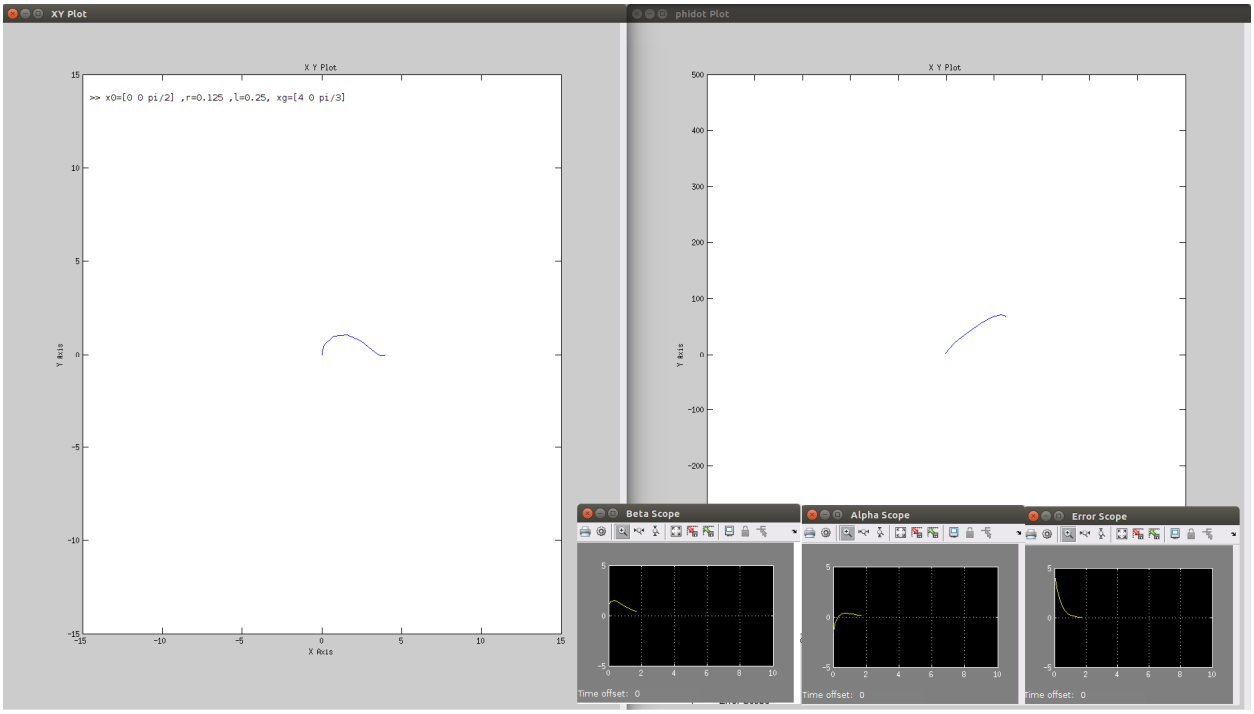


شکل 2-2: مدار کنترلی ربات دیفرانسیلی

آزمایش‌های زیادی جهت بررسی عملکرد مدار شکل شماره 2-2 انجام شده اند که نتایج دو نمونه از آنها در شکل‌های 2-3 و 2-4 آمده اند.



شکل 2-3: آزمایش شماره ی 1 برای مدار شکل 2-2



شکل 2-4: آزمایش شماره ی 2 برای مدا شکل 2-2

3 - پیاده سازی در نرم‌افزار Webot

در نرم‌افزار ویبوت مهم‌ترین قسمت پیاده سازی توابع Differential و gotoPointDiff هستند که به صورت کدهای 3-1 و 3-2 پیاده‌سازی شده اند.

```
// gets the speed of left and right wheel with their details and calculates the new location of the robot
// @param   left        The left wheel's angular velocity
// @param   right       The right wheel's angular velocity
// @param   r           The wheels' radius
// @param   l           The destance between 2 wheels
// @param   current     The current position of robot
// @param   dt          The time granity used for integral purposes
// @return  position_t  The new position of robot based on inputs
position_t differential(phidot_t left, phidot_t right, length_t r, length_t l, position_t* current, dtime_t dt){
    // define locals
    dot_t tdot, p, xdot, ydot;
    radian_t theta;
    // calc. the fix part of eq.
    phidot_t
        phi_sum = right + left,
        phi_sub = right - left;
    // if the pev. is null
    if(current == NULL){
        // assume all zero
        current = pos_new(0, 0, 0);
    }
    // calc. the \dot{\theta}
    tdot = (thetadot_t)(r * phi_sub / ( 2 * l ));
    // make the integral
    theta = tdot * dt  + current->theta;
    // make the fixed part of {x|y}dot eq.
    p = r * phi_sum / 2;
    // calc. the x dot
    xdot = p * (radian_t)cos(theta);
    // calc. the y dot
    ydot = p * (radian_t)sin(theta);
    // return an integrated new pos.
    return pos_make(xdot * dt + current->x, ydot * dt + current->y, __ANGDIFF(theta));
}
```

کد 3-1: پیاده سازی تابع Differential

که در تابع فوق مقدار dt به صورت ثابت با مقدار 0.001 است که با آزمون خطا بدست آمده است زیرا که در نرم‌افزار ویبوت بخاطر باید مقدار dt و قدم های زمانی ربات به اندازه‌ای انتخاب شوند که ربات فرصت این را پیدا کند تا بتواند با تغییراتی که در سرعت چرخ‌هایش ایجاد میکند مسیر خود را تنظیم کند در غیر این صورت ربات از لحاظ محاسباتی به نقطه‌ی مورد نظر همگرا خواهد شد ولی از منظر شبیه‌ساز به جایی نمیرسد.

```
14  #define ERROR_THRESHOLD     0.05f
15  #define K_RHO               +3.0f
16  #define K_ALPHA             +8.0f
17  #define K_BETA              -1.5f
18
19  // linear and angular velocity configuration container
20  typedef struct vw{
21      float v;
22      float w;
23  } vw_t;
24  // returns a new linear and angular velocities based on
25  // @param  p    current position of robot
26  // @param  pg   target position of robot
27  // @output STOP has the robot reached the target destination?
28  // @return vw_t a new velocity configuration
29  vw_t gotoPointDiff(position_t p, position_t pg, bool* STOP) {
30      vw_t $;
31      length_t rho    = pos_calc_destance(p, pg);
32      float __atan2   = atan2f(pg.y - p.y, pg.x - p.x);
33      length_t beta   = pg.theta -__atan2;
34      length_t alpha  = __atan2 - p.theta;
35      *STOP           = (rho <= ERROR_THRESHOLD);
36      $.v             = K_RHO * rho;
37      $.w             = K_ALPHA * alpha + K_BETA * beta;
38      return $;
39  }
40
41  #undef ERROR_THRESHOLD
42  #undef K_RHO
43  #undef K_ALPHA
44  #undef K_BETA
45
```

کد 3-2: پیاده سازی تابع gotoPointDiff

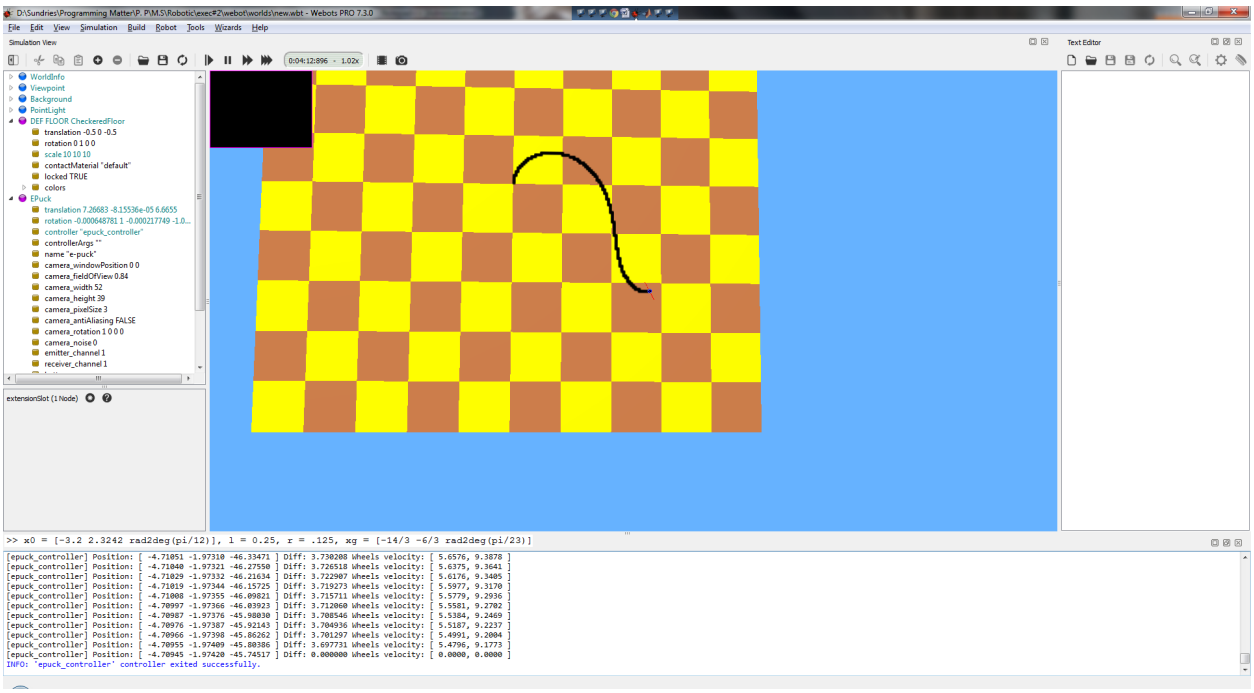
و قسمتی که این توابع کدهای 3-1 و 3-2 را باهم به‌کار میگیرد و در شبیه‌ساز به اجرا می‌آورد در کد 3-3 آمده است.

```
    wb_robot_init();
    {
        // get the pen device
        pen = wb_robot_get_device("pen");
        // init x0 and xg
        init_pos(&x0, &xg);
        // define the initial position as current position
        x = x0;
        // drive the robot
        while(!STOP && wb_robot_step(TIME_STEP) != -1) {
            // trace the path
            wb_pen_write(pen, true);
            // print current position
            printf("Position: ");
            __PRINT_POSITION(stdout, x);
            // calc the new phi-dots value
            get_phidots(gotoPointDiff(x, xg, &STOP), (length_t)WHEEL_RADIUS_CM / 100, (length_t)ROBOT_RADIUS_CM / 100, &phildot, &phirdot);
            // if should stop?
            if(STOP) { phildot = 0; phirdot = 0; }
            // normalize the velocities
            normalize_velocities(&phildot, &phirdot);
            // print status
            printf("\t\tWheels velocity: [ %.4f, %.4f ]\n", phildot, phirdot);
            // set the wheels' velocity
            wb_differential_wheels_set_speed(phildot, phirdot);
            // get the new location of the robot
            x = differential(phildot, phirdot, (length_t)WHEEL_RADIUS_CM/100, (length_t)ROBOT_RADIUS_CM/100, &x, DT);
        }
        // stop tracing the path
        wb_pen_write(pen, false);
        printf("Execution terminated ...");
    }
    wb_robot_cleanup();
```

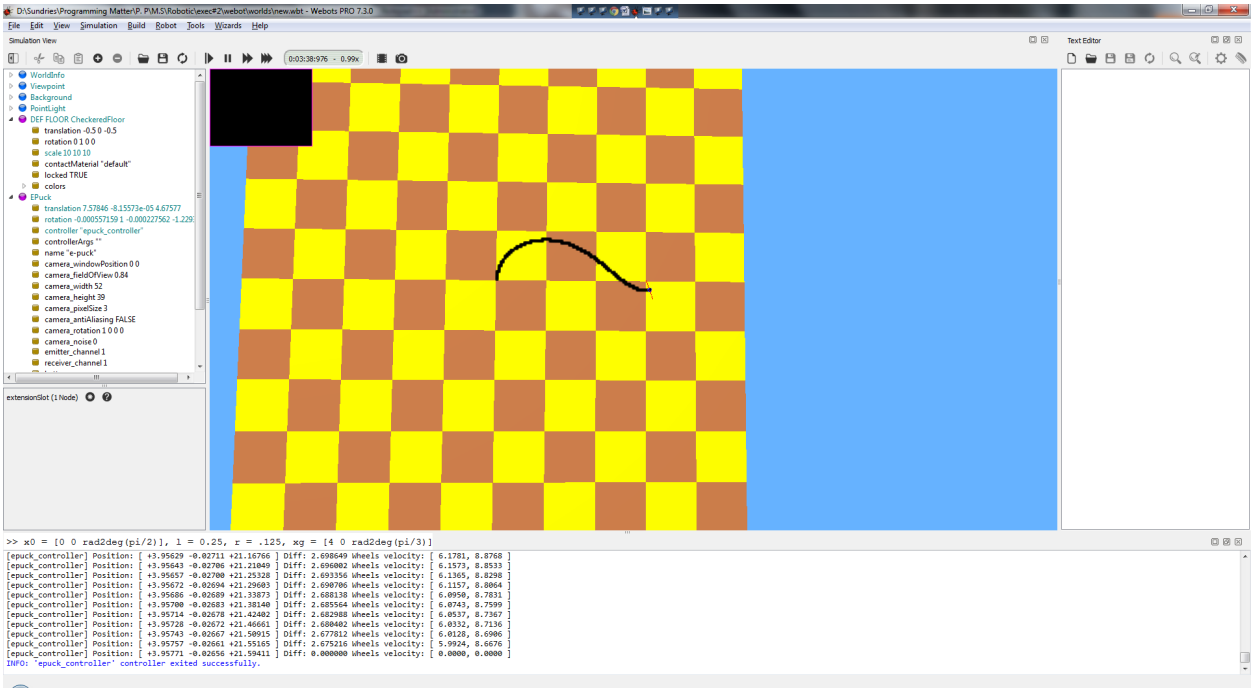
کد 3-3: ترکیب توابع اشکال 3-1 و 3-2 و شبیه‌ساز

که توضیح مفصل راجع به دیگر اجزا کد ها در ارائه حضوری صورت خواهد گرفت.

جهت بررسی اینکه آیا معادلات(1)تا(16) به درستی در شبیه‌ساز پیاده‌سازی شده اند ما آزمایش‌های شکل‌های 3-2 و 4-2 را در شبیه‌ساز اجرا کرده این که در اشکال 3-1 و 3-2 نمایش داده شده اند.



شکل 3-1: اجرای آزمایش 3-2 در شبیه‌ساز



شکل 3-2: اجرای آزمایش 2-4 در شبیه ساز

فقط توجه داشته باشیم که ربات در شبیه‌ساز همیشه فرض را بر این میگذارد که در موقعیت شروع واقع است برای همین شکل 3-1 دوران یافته‌ی شکل 3-2 میباشد(به دلیل وجود زاویه اولیه مخالف 90 درجه) و همانطور که نتایج نشان میدهند توانسته‌ایم به درستی مدارهای شماره 1-1 و 2-2 را در شبیه‌ساز پیاده‌سازی کنیم و به اجرا آوریم.[2]

[2] نسخه ی بروز تمرین انجام شده را میتوانید از آدرس زیر دسترسی پیدا کنید که لینک داده شده تا روز ارائه‌ی حضوری معتبر خواهد بود.
https://github.com/noise2/tmp

4 - منابع

- اسلایدهای درس
- کتاب مرجع اصلی - نوربخش
- http://planning.cs.uiuc.edu/node659.html
- https://www.youtube.com/watch?v=aE7RQNhwnPQ