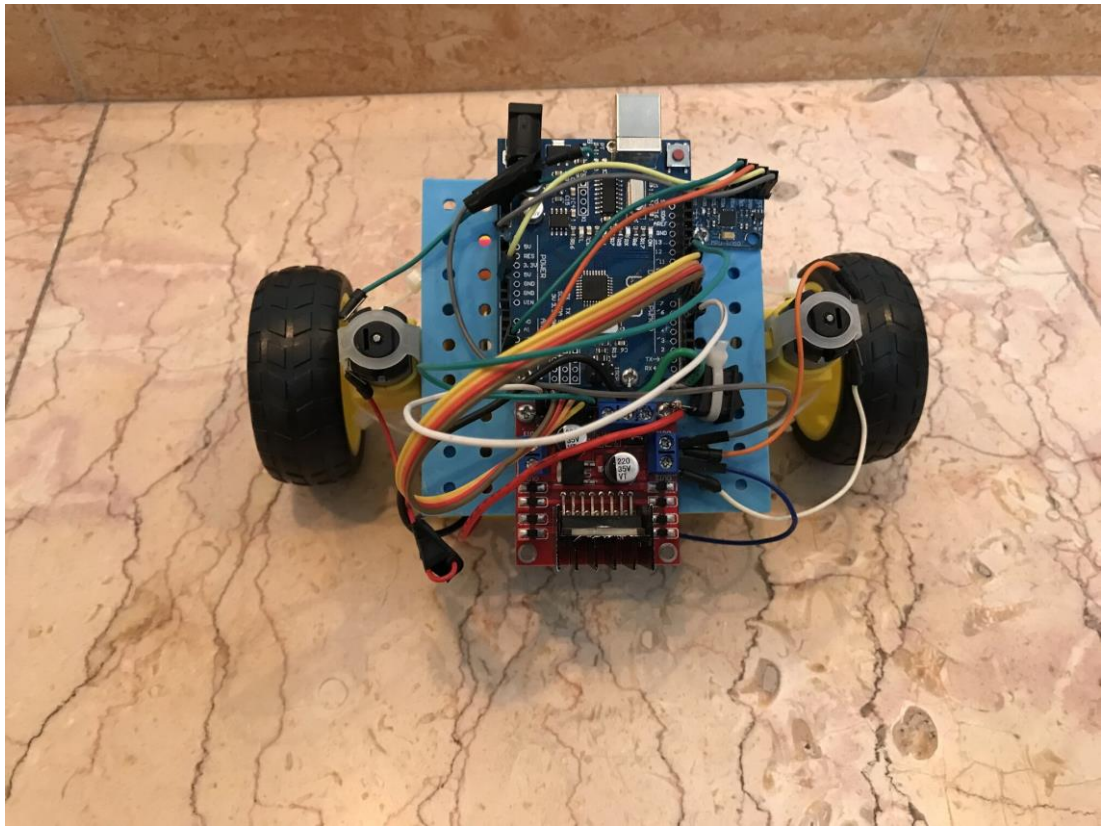
کنترل ربات تعادلی با رویکرد LQR

Alireza Abassi

هدف

- در این ارائه قصد دارم کنترل ربات سلف بالانس را انجام دهم.



نمونه مشابه

- برای انتخاب کنترلر بهینه (PID – LQR – P-LQR) از مقاله زیر استفاده کرده‌ام.

Enhanced Longitudinal Motion Control of UAV Simulation by Using P-LQR Method

Kok Kai Yit[1] and Parvathy Rajendran[2*]
School of Aerospace Engineering, Universiti Sains Malaysia, Engineering Campus,
14300 Nibong Tebal, Pulau Pinang, Malaysia
[1]kok901221kaiyit@yahoo.com, [2]aeparvathy@usm.my

Models of PID and LQR

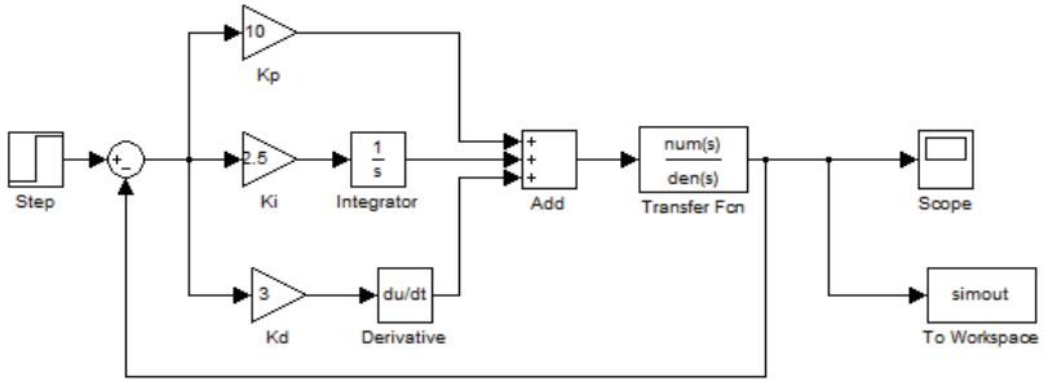


Figure 1: Simulink model of PID controller.

can be determined in MATLAB by using command lqr (A, B, C, D, Q, and R) [17-18].

$$K=[2.405 \quad 4.700 \quad 0.377] \tag{8}$$

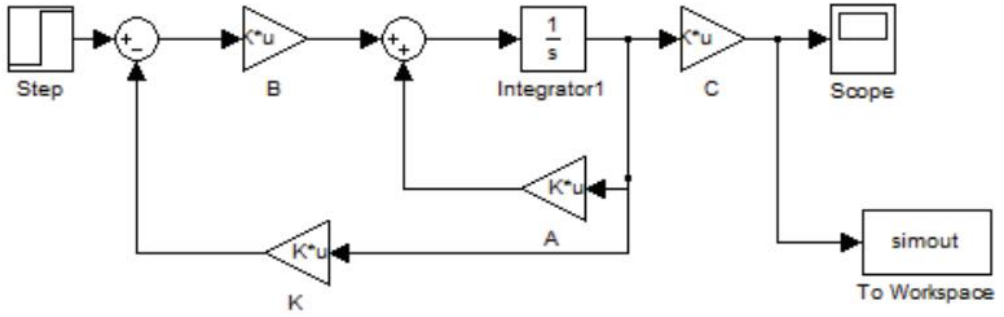


Figure 2: Simulink model of LQR controller.

P-LQR model

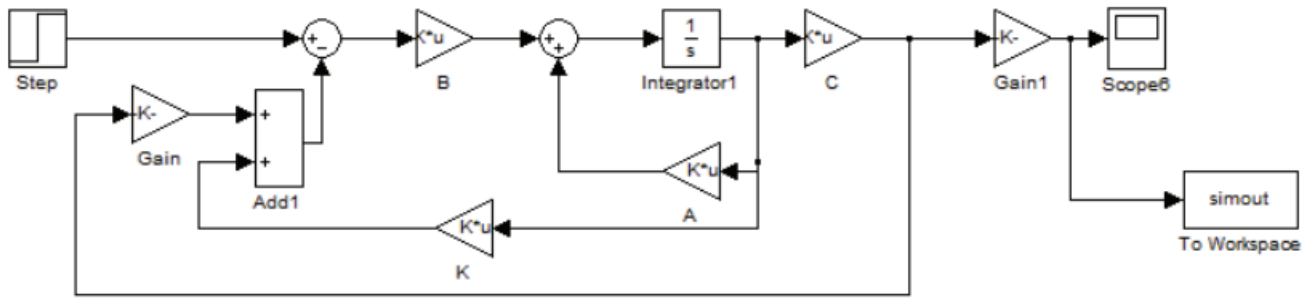


Figure 3: Simulink model of the P-LQR controller.

The addition gain in the close loop is shown as "Gain" while the balancing purpose gain is shown as "Gain1" in the figure. The calculation of the balancing purpose gain can be done by using eqn. 9.

$$K_{Gain1} = 1 + K_{Gain} \tag{9}$$

Comparisons between The Controllers in an UAV example

Table 1: Comparison of various parameters among the PID, LQR, and P-LQR controllers

Control Algorithms	PID	LQR	P-LQR
Rise Time (s)	0.622	0.196	0.111
Settling Time (s)	1.161	0.330	0.202
RMSE	0.253	0.135	0.113
Overshoot (%)	0.093	0.920	8.100
Steady state error	0.001	0.002	0.001

انتخاب رویکرد کنترلر

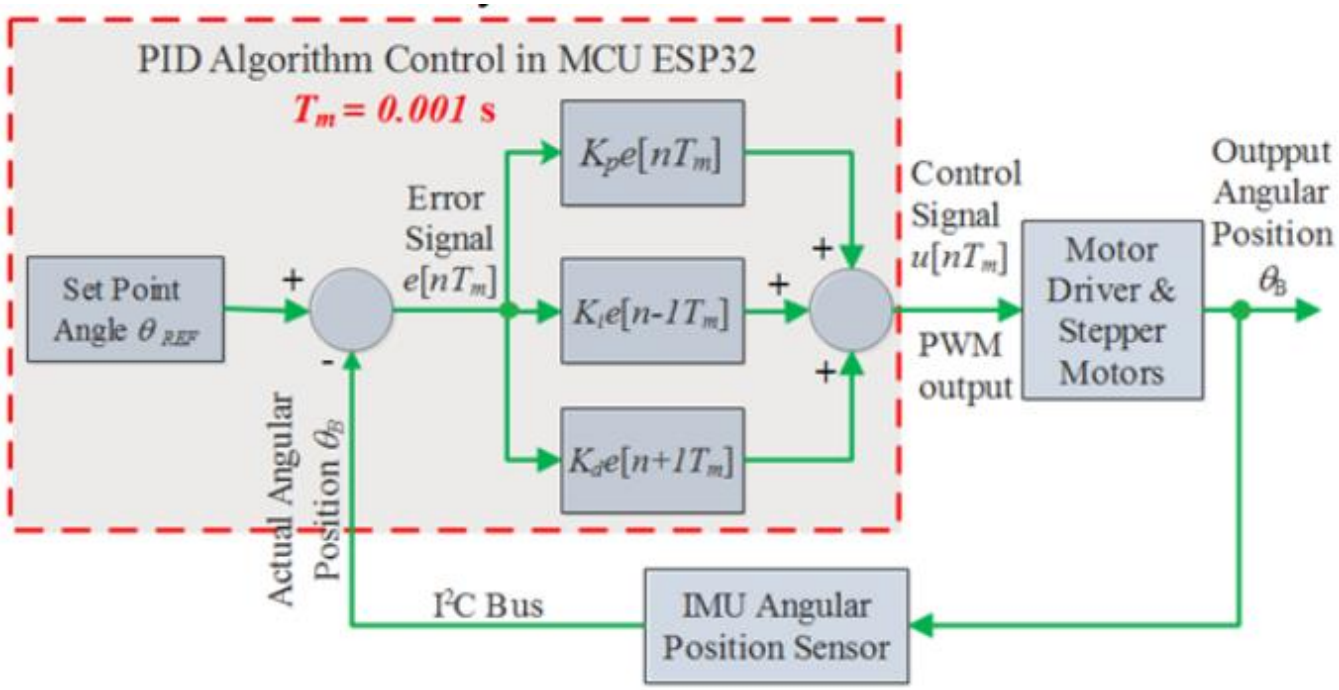
- از مقایسه ۳ کنترلر متوجه میشویم که در این مساله P-LQR بهتر از LQR و آن هم بهتر از PID بوده است. پس در این کاربرد، بنابر سادگی ابتدا کنترلر PID را پیاده‌سازی میکنیم. در گام بعد کنترلر LQR و در نهایت با تلفیق آن دو، هدف آن است که کنترلر P-LQR را پیاده سازی کنیم.
- برای پیاده سازی کنترلرها روی برد آردوینو از مقاله زیر کمک گرفته‌ام.
- از آنجا که رویکرد مقاله آموزش-محور است، بسیار در پیاده‌سازی کمک‌کننده است.

Modeling and Control of a Two Wheeled Auto-Balancing Robot: a didactic platform for control engineering education

Fabián R. Jiménez L., MSc.[1], Ilber A. Ruge R., MSc. [2], and Andrés F. Jiménez L., PhD [3]
[1,2] Pedagogical and Technological University of Colombia - Department of Electronic Engineering - I2E Research Group, Tunja, Colombia. {fabian.jimenez02, ilber.ruge}@uptc.edu.co, 3Universidad de los Llanos - Faculty of Sciences and Engineering, Villavicencio, Colombia, ajimenez@unillanos.edu.co.

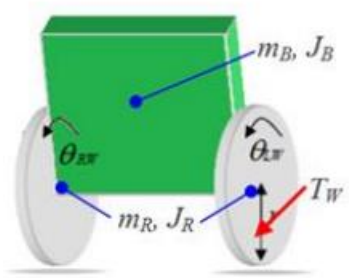
کنترلر PID

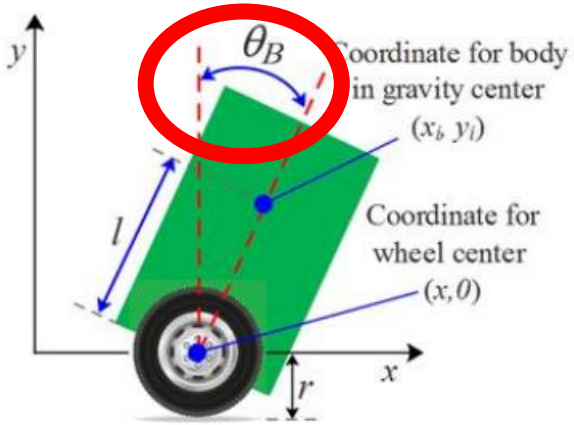
- از بلوک دیاگرام زیر استفاده میکنیم.



سنسور

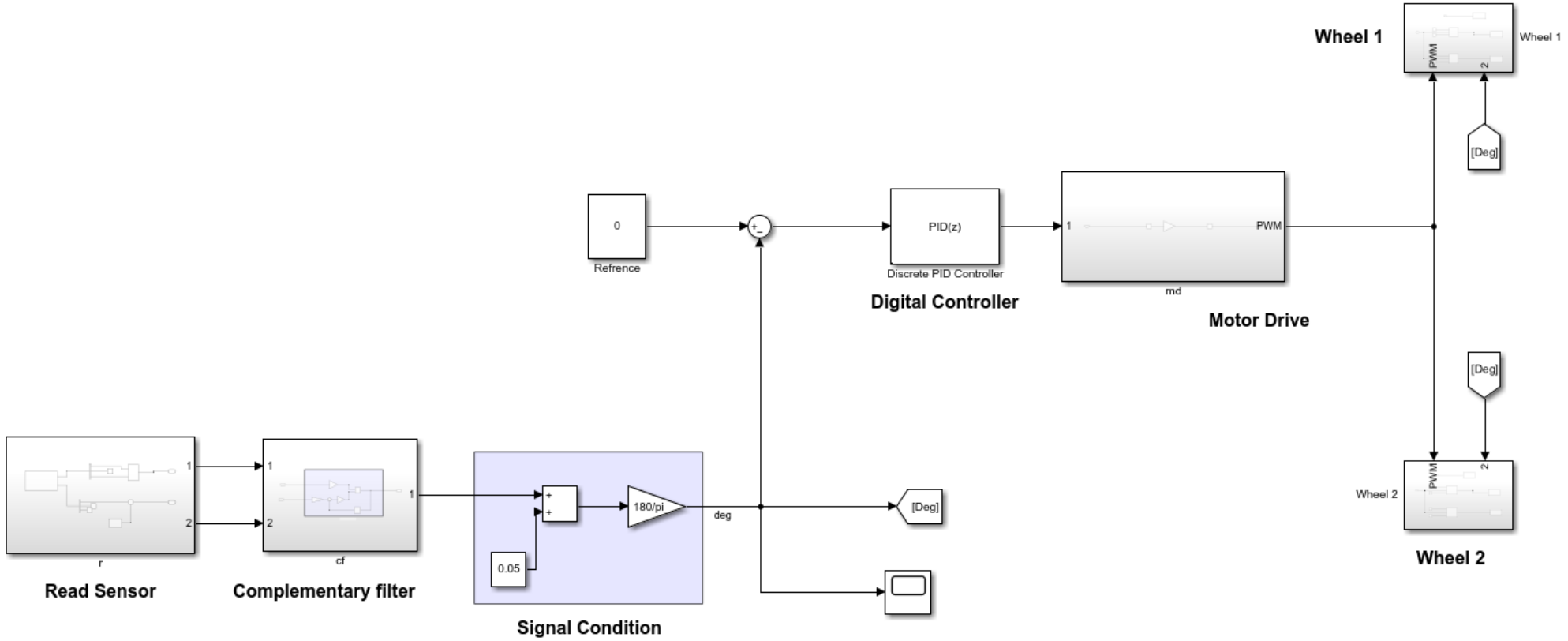
- از سنسور یا مشاهده‌گر MPU6050 استفاده میکنم که زاویه پیچ ربات را میدهد.
- بدیهیست برای حالاتی که در دسترس نداریم برای کنترلر LQR باید از تخمینگر استفاده شود.





پیاده‌سازی PID

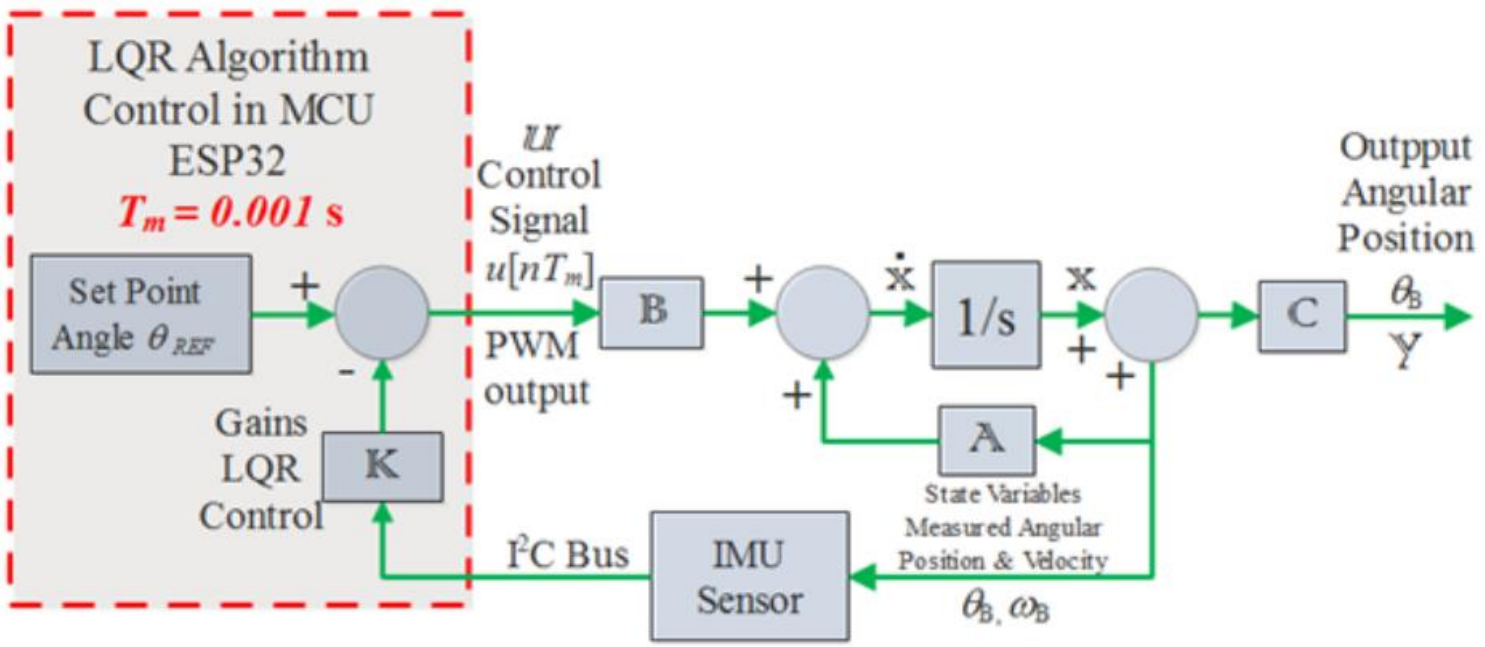
- بیش از این برای این کنترلر در این ارائه وقت نمیگذاریم و کنترلر نهایی را می‌بینیم.



نتیجه PID

- کنترل ربات با رویکرد PID به سادگی صورت گرفت. زیرا برای ورودی و خروجی گزینه دیگری نداشتیم و کنترلر نیز تنها ۳ ضریب برای تنظیم دارد. که بصورت Real-time میتواند تنظیم شود.

کنترلر LQR


LQR Algorithm
Control in MCU
ESP32
$T_m = 0.001$ s
Set Point
Angle θ_{REF}
+
-
Gains
LQR
Control
K
ur
Control
Signal
$u[nT_m]$
PWM
output
B
+
+
$\dot{x}$
1/s
x
+
+
C
Outpput
Angular
Position
θ_B
y
A
State Variables
Measured Angular
Position & Velocity
θ_B, ω_B
I²C Bus
IMU
Sensor

CAE IN DER REGELUNGSTECHNIK: EXERCISES

General Problem Setting

The *Pendulum-Cart* system defines a classic problem in dynamics and control theory and is widely used as a benchmark for testing control algorithms. A simplified variation of this system is illustrated in Figure 2 where a mathematical pendulum is connected to a pair of only-rolling wheels with one translation degree of freedom. This models the *Wheeled Inverted Pendulum* system. An example for that is the Robot *Suricate* (Figure 1) built by the students at LRS / University of Kaiserslautern.

In the simplified model, the point mass m_p is connected to the cart with the mass m_c via a massless arm with the length L. q_1 is the displacement of the cart and q_2 is the angle of the pendulum. As input, u is assumed to be a force.

Abbildung 1: Suricate (JIM2)

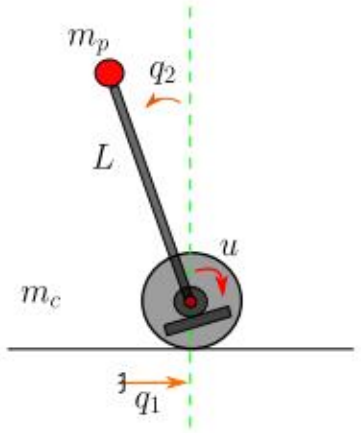
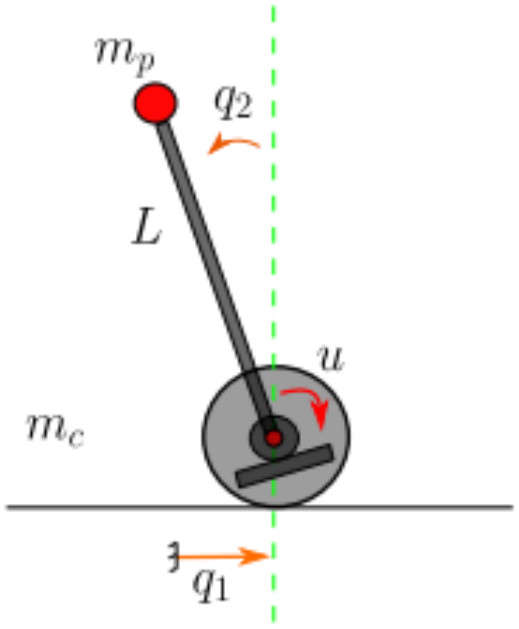


Abbildung 2: Pendulum-Cart System

فضای حالت

- برای پیاده‌سازی این کنترلر از آنجا که همه حالات را نمیتوانیم با سنسور اندازه بگیریم، نیاز به تخمینگر داریم. و برای پیاده‌سازی تخمینگر نیز نیاز به معادلات دینامیک سیستم داریم .
- برای پیاده‌سازی این کنترلر بعلت سادگی از فایل سمت چپ استفاده کرده‌ام. که ویدئوی شبیه‌سازی آن نیز در یوتیوب موجود است.

معادلات فضای حالت



- مدل درنظر گرفته شده، تقریبا همان پاندول معکوس می‌باشد.

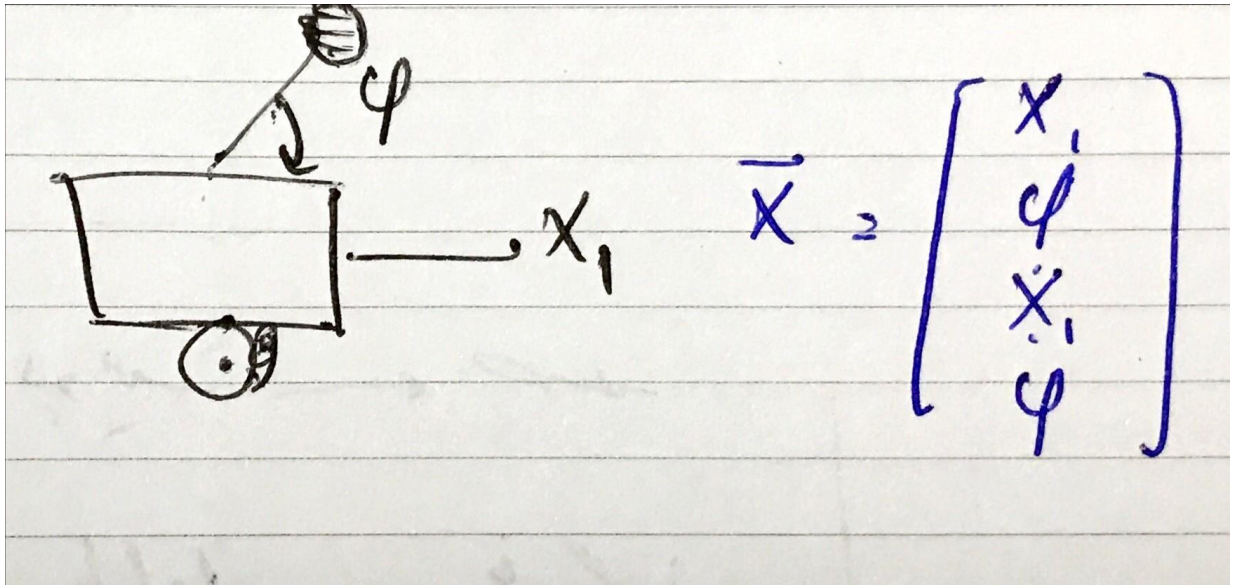
$$\boldsymbol{A} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{g\,m_p}{m_c} & -\frac{d_1}{m_c} & -\frac{d_2}{L\,m_c} \\ 0 & \frac{g\,(m_c+m_p)}{L\,m_c} & -\frac{d_1}{L\,m_c} & -\frac{d_2\,(m_c+m_p)}{L^2\,m_c\,m_p} \end{pmatrix}, \boldsymbol{B} = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{m_c} \\ \frac{1}{L\,m_c} \end{pmatrix}.$$

- از آنجا که سنسور mpu زاویه را به ما میدهد. بانبراین ماتریس خروجی را چنین درنظر میگیرم:

```
C = [0,  1,  0,  0;];
```

متغیر های حالت

- متغیر های حالت بصورت زیر میباشند.



طراحی کنترلر LQR

- باید برای plant در دسترس یک کنترلر LQR طراحی کنیم. دینامیک plant را وارد فضای حالت میکنیم:
- پارامترها را وارد میکنیم:

```
%% Define the parameters

mc = 1;
mp = 0.5;
g = 9.82;
l = 0.1;
d = 0.01;
```

طراحی کنترلر LQR

- فضای حالت(ماتریس‌ها) را پیاده‌سازی میکنیم:

```
%% Define matrice

A = [0,              0,           1,           0;
     0,              0,           0,           1;
     0,      (g*mp)/mc,      -d/mc, -d/(l*mc);
     0, g*(mc+mp)/(l*mc),-d/(l*mc),    -d*(mc+mp)/(l^2*mc*mp)];
B = [0;
     0;
     1/mc;
     1/(l*mc)];
C = [0,  1,  0,  0;];
D = 0;
```

```
  %% Build system

sys = ss(A, B, C, D);
```

طراحی کنترلر LQR

- از آنجا که قصد پیاده‌سازی روی یک کنترلر دیجیتال را داریم، باید فضای حالت را سمپل کنیم:

```
%% Discrete Time
sys_d = c2d(sys, 0.01);
Ad = sys_d.a;
Bd = sys_d.b;
Cd = sys_d.c;
Dd = sys_d.d;
```

طراحی کنترلر LQR

- با انتخاب پارامترهای Q و R، بهره دیجیتال LQR را بدست می‌آوریم:

```
%% controller

Q = eye(4);
R = 1;

K = lqrd(Ad, Bd, Q, R, 0.01);
```

تخمین حالات

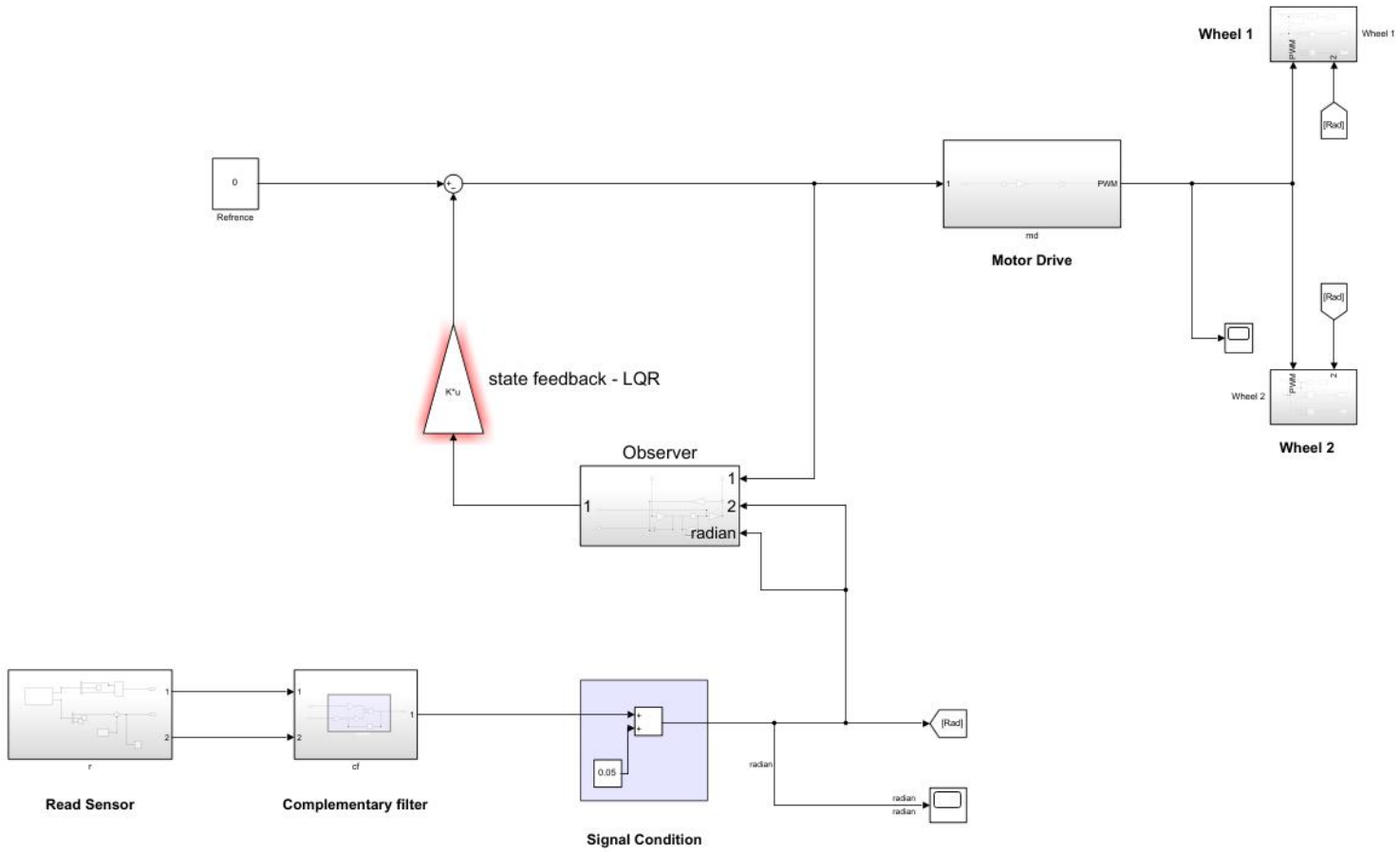
- برای حالاتی که در دسترس نداریم باید تخمینگر کاهش مرتبه طراحی کنیم. بدیهیست چون این مدل بطور کامل مشاهده پذیر نیست نمیتوانیم از تخمینگر مرتبه کامل استفاده کنیم. همچنین باید محل قطب های تخمینگر را بسیار دورتر از قطب‌های plant درنظر بگیریم. زیرا باید تخمینگر سریعتر از خود سیستم عمل کند.برای این مهم در فضای حالت دیجیتال، باید قطب‌ها را در دایره صفرتا یک، به مبدا نزدیکتر کنیم.

تخمین حالات

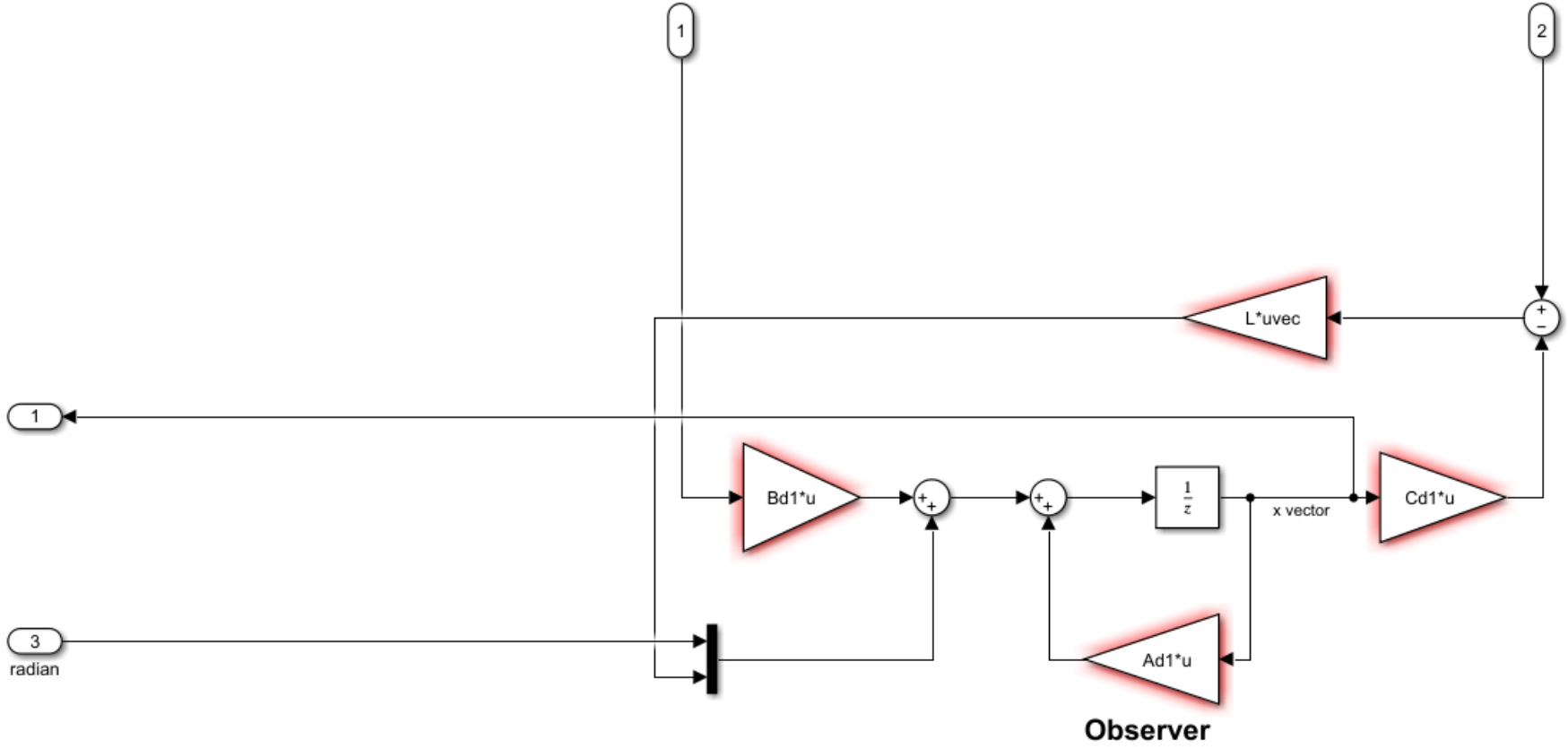
```
P = [Cd;
     0,  0,  1,  0;
     1,  0,  0,  0;
     0,  0,  0,  1];
Ad1 = P*Ad/P;
Bd1 = P*Bd;
Cd1 = C/P;
Ad12 = Ad1([1], [1 2 3]);
Ad22 = Ad1([2 3 4], [2 3 4]);

des_pole_od = (eig(Ad1))/10;
L = acker(Ad22', Ad12', des_pole_od([1, 3, 4]))';
```

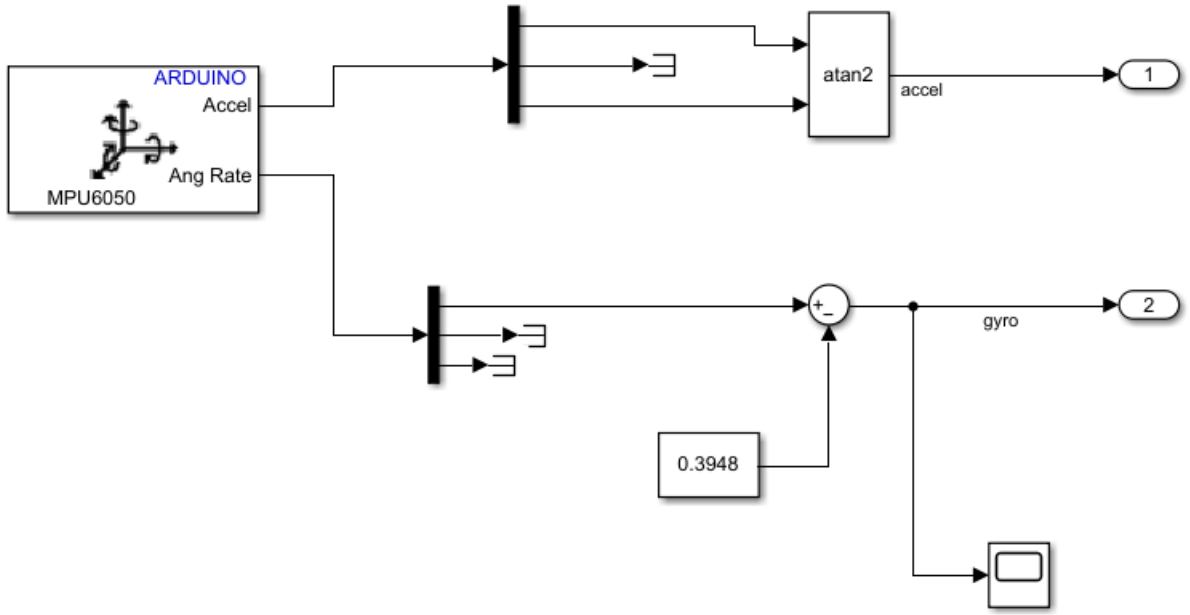
سیمولینک و اتصال به دنیای واقعی



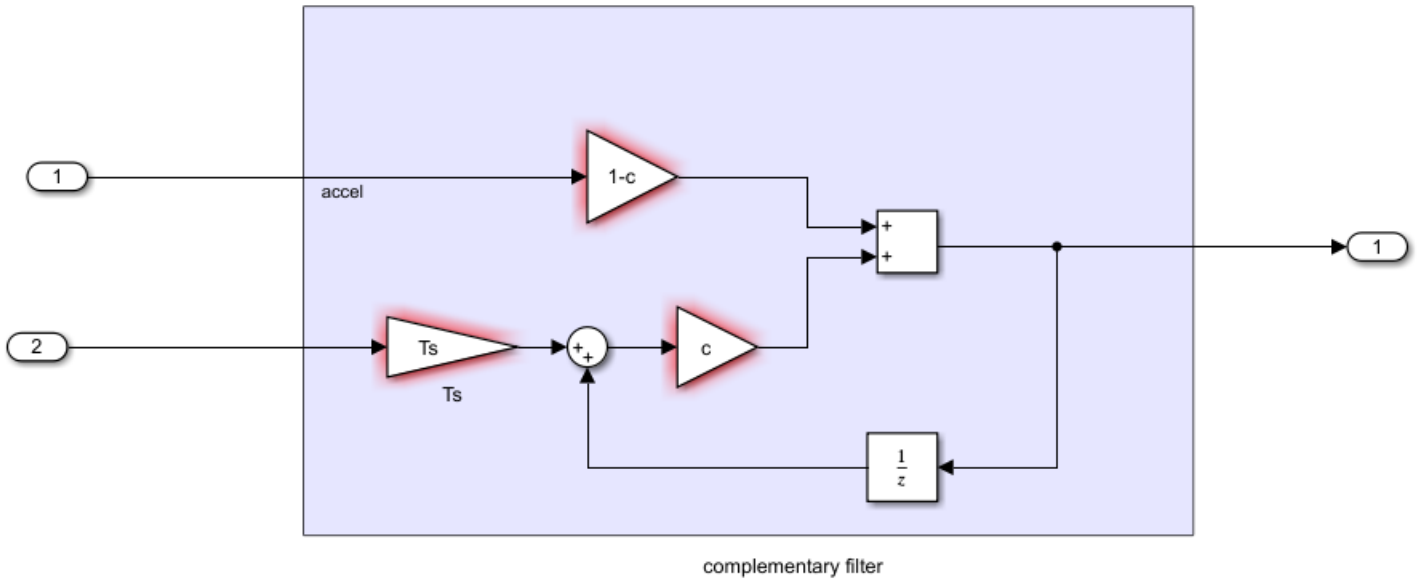
بلوک دیاگرام تخمینگر



بلوک دیاگرام سنسور

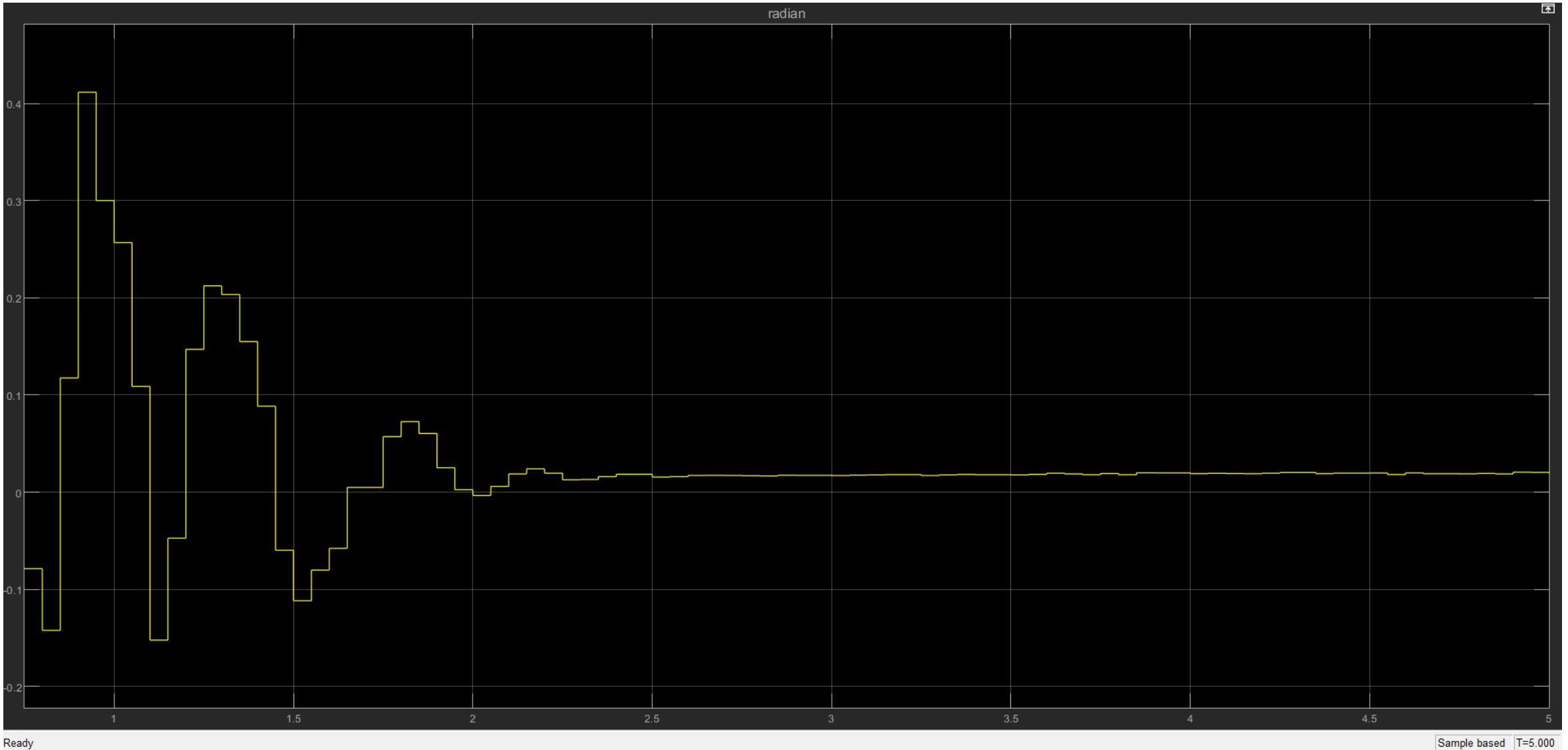


بلوک دیاگرام فیلتر مکمل



چند تلاش موفق از ربات برای حفظ تعادل

پاسخ سیستم برای دفع اغتشاش در دنیای واقعی



مثالی دیگر از پاسخ سیستم