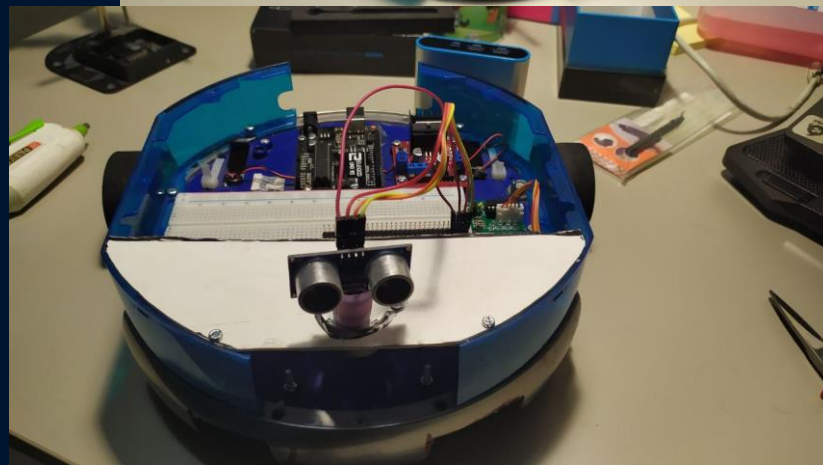
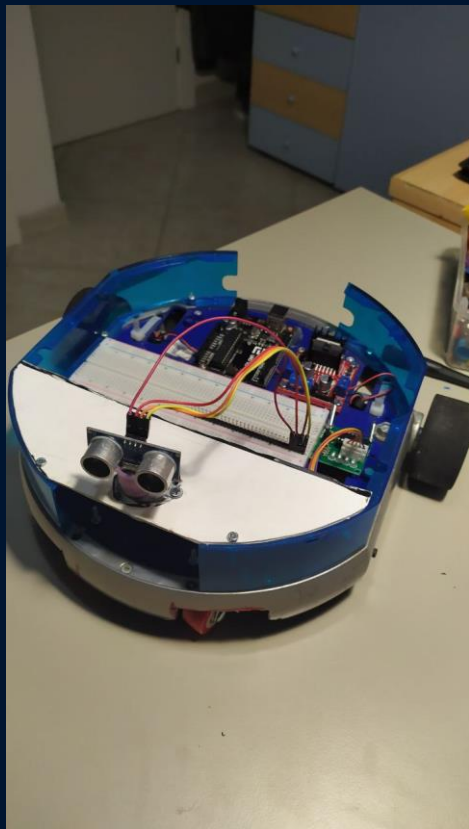


Rover Inseguitore

*Lijoi Letizia
Sabellico Danilo
Chiarucci David*

Foto





01

**Modello
cinematico**

Analisi
Matematica
Della dinamica
Del sistema

02

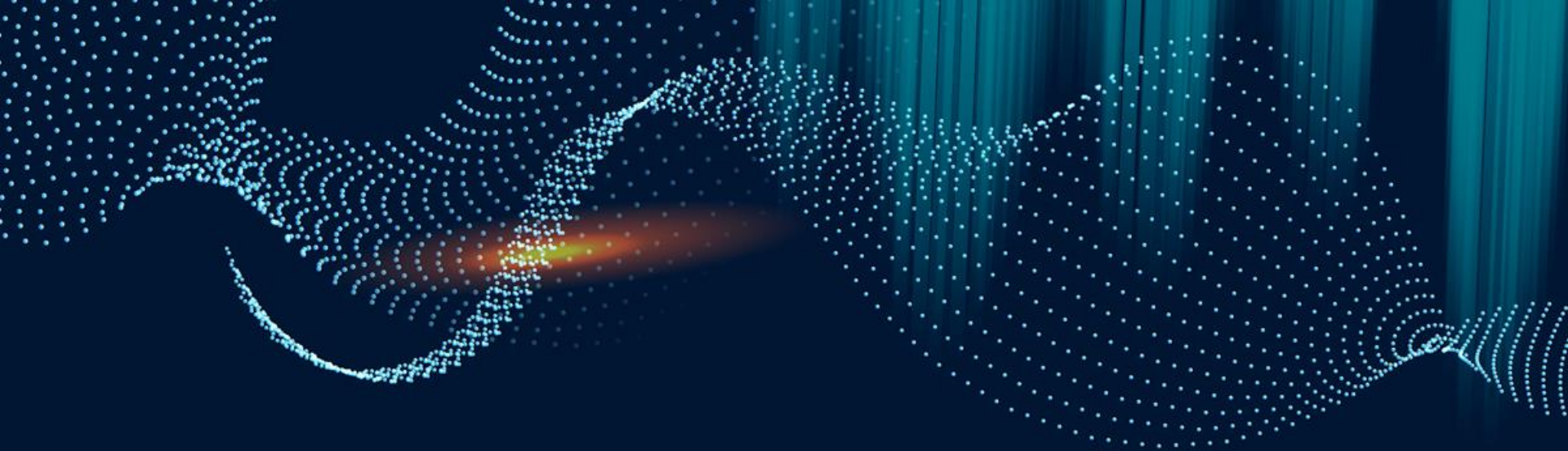
**Legge di
controllo**

Descrizione
della legge di controllo
adottata per la realizzazione del
progetto

03

**Funzione di
trasferimento**

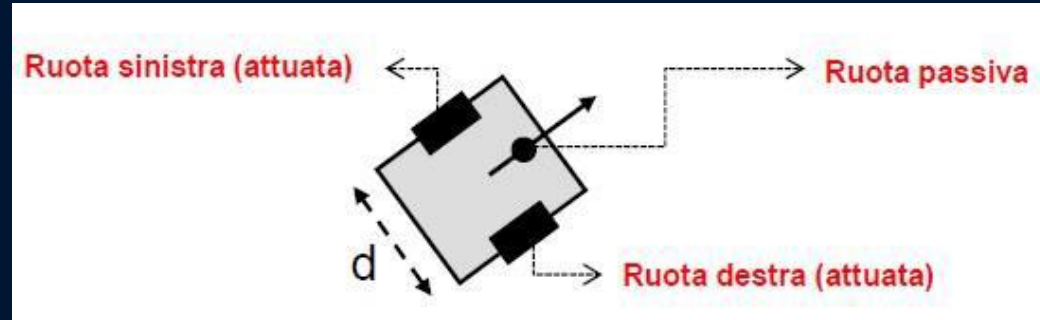
Analisi
dello schema a
blocchi e
descrizione della
dinamica del
sistema



01 | **Modello cinematico**

Cinematica unicycle

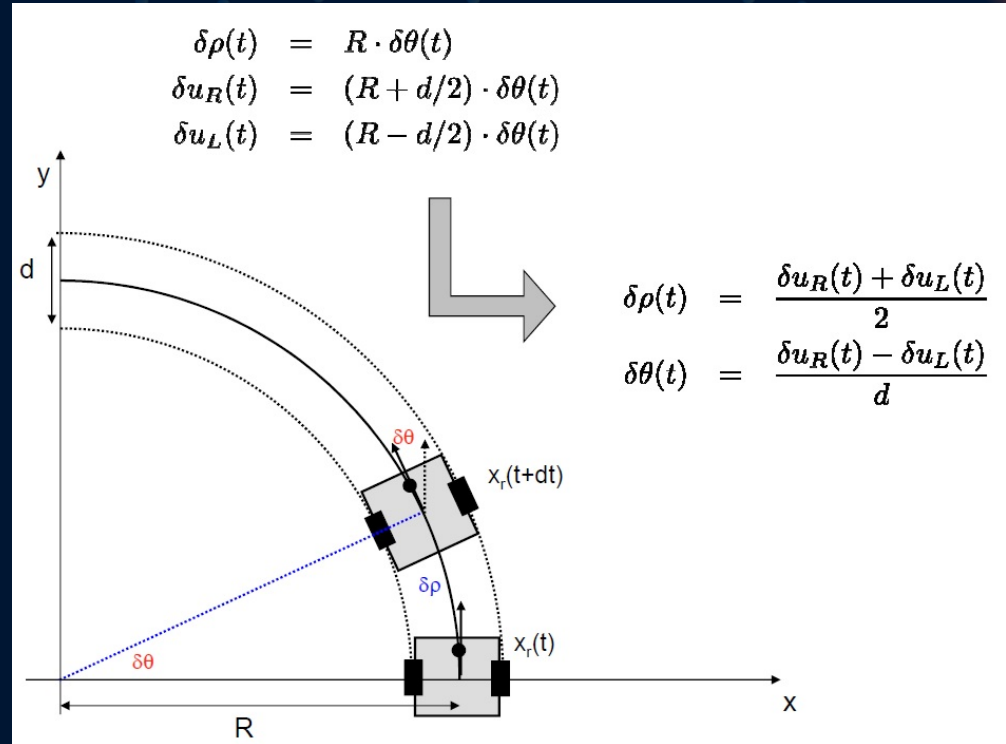
- L'unicycle da noi realizzato ha una struttura a guida differenziale
- La posa dell'unicycle è data dalle variabili x e y , che individuano la posizione, e dall'angolo θ , che definisce l'orientamento.
- $q = [x, y, \theta]'$



Cinematica Uniciclo

- Leggi cinematiche:

$$\begin{aligned}\dot{x} &= \frac{v_R + v_L}{2} \cos(\theta) \\ \dot{y} &= \frac{v_R + v_L}{2} \sin(\theta) \\ \dot{\theta} &= \frac{v_R - v_L}{d}\end{aligned}$$





02 | Legge di controllo

Legge proporzionale nelle velocità

Le variabili di controllo sono: la velocità longitudinale v_1 , proporzionale alla distanza dell'uniciclo dal target; la velocità rotazionale v_2 , proporzionale alla differenza tra l'orientazione dell'uniciclo e il target.

- In formule:

$$v_1 = K_{v1} * e_p * \cos \varphi \quad (1)$$

$$v_2 = K_{v2} * (\theta_d - \theta) \quad (2)$$

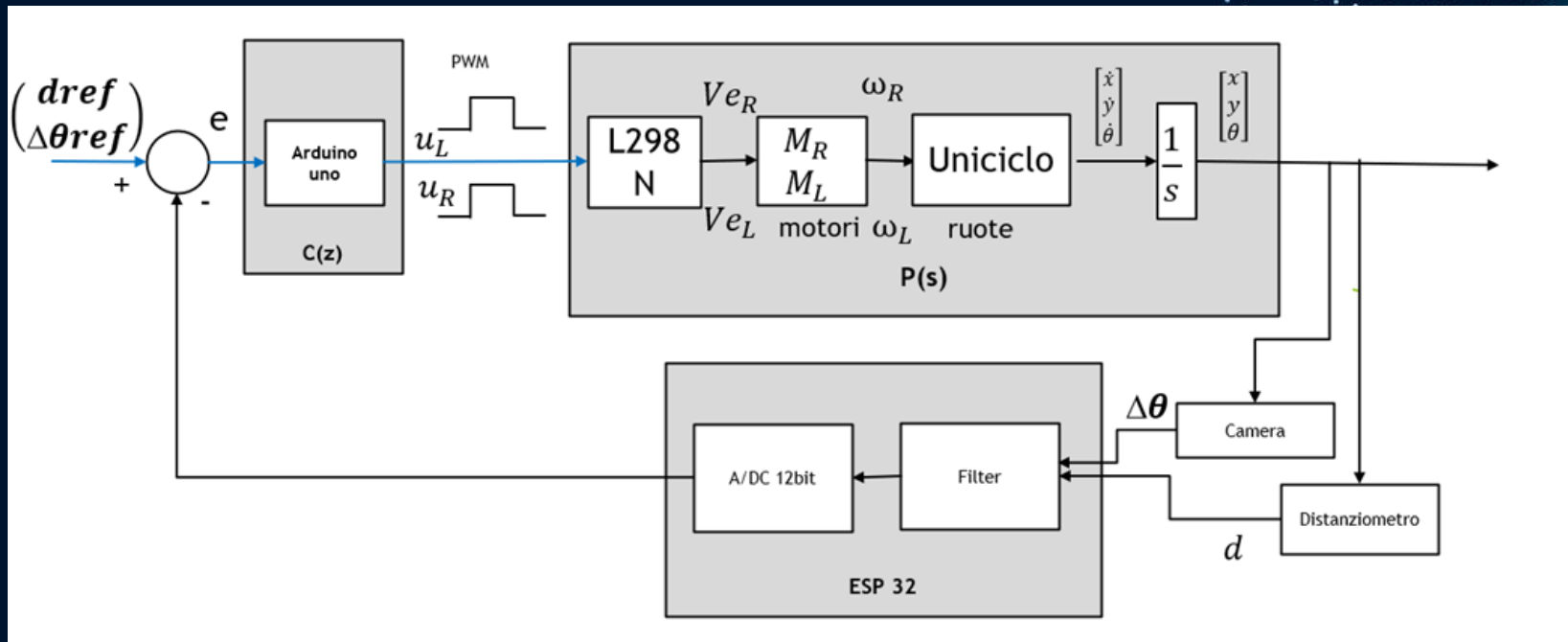
- Linearizzazione della relazione (1).

$$v_1 = K_{v1} * x \quad (3)$$



03 | **Funzione di trasferimento**

Schema a blocchi



Blocco P(s)

Meccanica

- Equazione di equilibrio ai momenti

$$J\dot{\omega} = \tau_m - c \cdot \omega \quad (4)$$

- La coppia applicata è proporzionale alla corrente

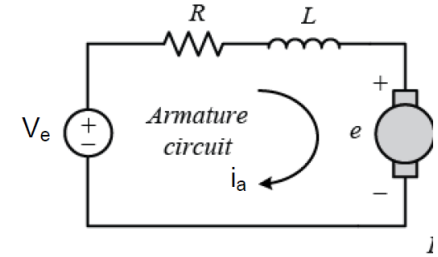
$$J\dot{\omega} = K_m i_a - c \cdot \omega \quad (5)$$

- Applichiamo Laplace

$$Js\omega(s) = K_m i_a(s) - c \cdot \omega(s) \quad (6)$$

$$\omega(s) = \frac{K_m}{Js+c} i_a \quad (7)$$

Elettrica



- Comportamento motore

$$V_e = Ri_a + L \frac{di_a}{dt} - E \quad (8)$$

- Essendo $E = K_e \omega$, applicando Laplace, si ha

$$V_e(s) = Ri_a(s) + Lsi_a(s) - K_e \omega(s) \quad (9)$$

- Conoscendo $\omega(s)$, ricaviamo

$$i_a(s) = \frac{Js+c}{s^2LJ+s(RJ+Lc)-K_eK_m} V_e \quad (10)$$

- Approssimando $i_a(s) \cong \alpha V_e(s)$, otteniamo

$$\omega(s) = \frac{K_m \alpha}{Js+c} V_e \quad (11)$$

Blocco P(s)

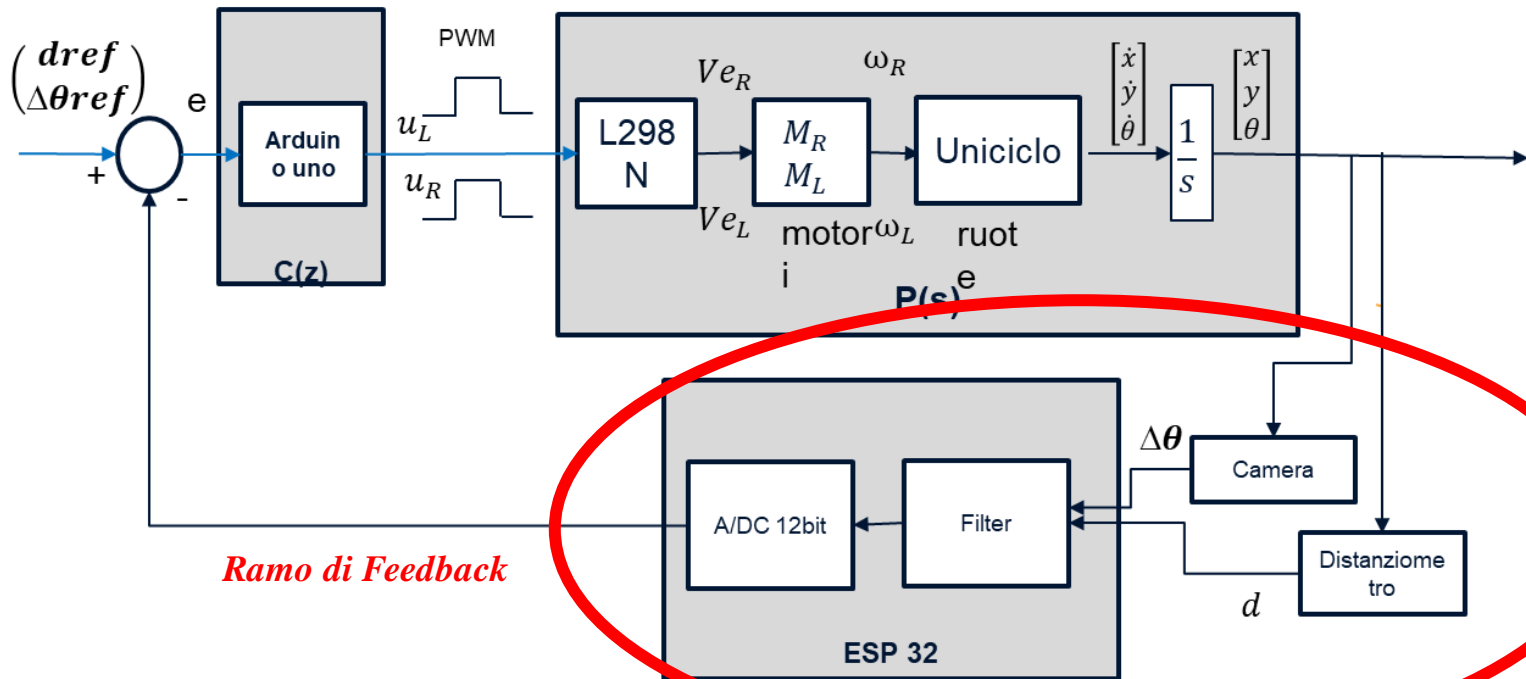
$$P(s) = \frac{K_m \alpha}{s^* (Js + c)} \quad (12)$$

- Fase di ricerca del target, la P(S) dovrà essere moltiplicata per $2*r/d$.

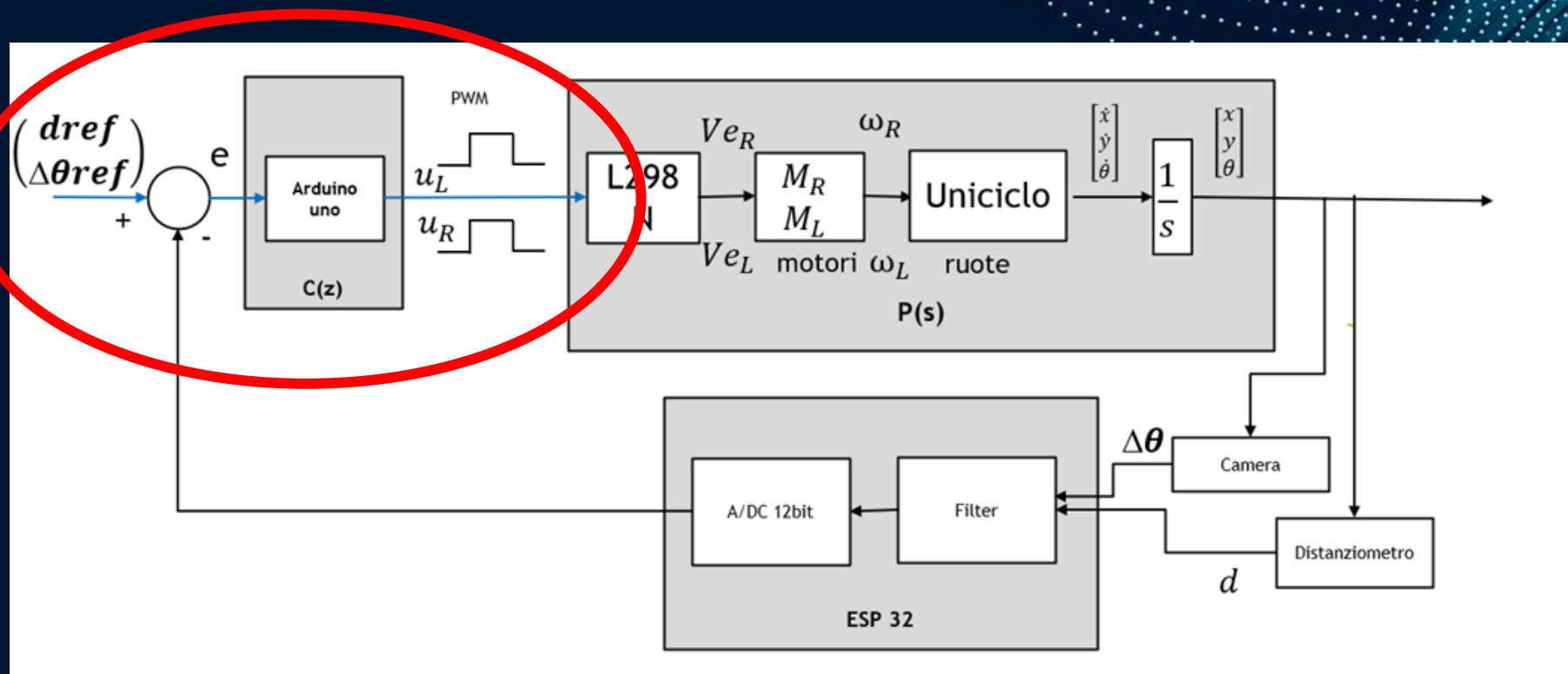
$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{(\omega_R + \omega_L)r}{2} \cos(\theta) \\ \frac{(\omega_R + \omega_L)r}{2} \sin(\theta) \\ \frac{(\omega_R - \omega_L)r}{d} \end{bmatrix} \quad (13)$$

- Fase di avvicinamento, la P(s) andrà moltiplicata per r

Ramo di feedback



Controllore



04

Hardware

Schema hardware e
lista dei principali
componenti
installati

05

Cenni sul codice

Cenni su
inizializzazione ed
esecuzione del
codice

06

Filtri Digitali

Maggiori filtri Digitali
implementati nel
progetto

07

Ritardi e campioni

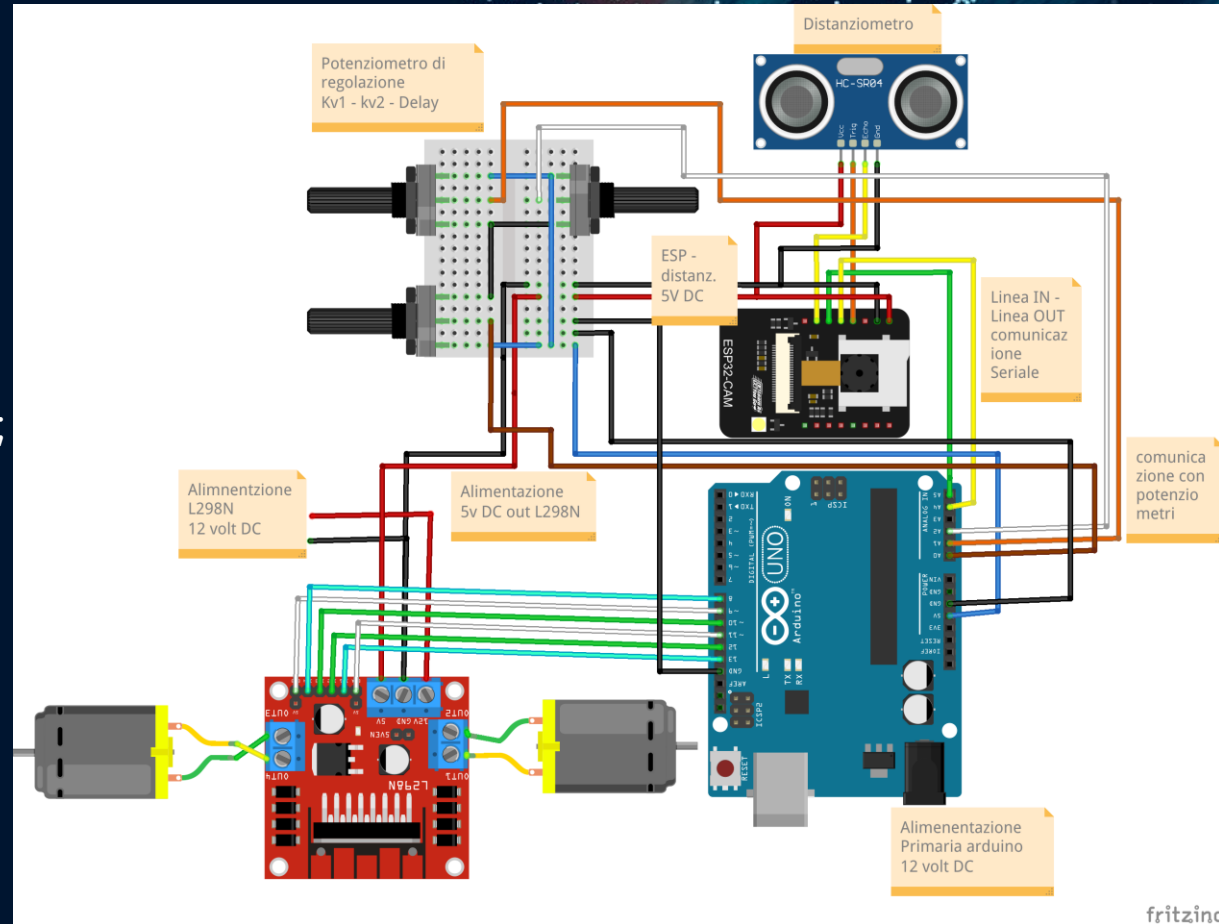
Ritardi di
campionamento
E
Trasmissione dei
campioni




04 | **Hardware**

Hardware

- Distanziometro;
- ESP32-CAM;
- ARDUINO;
- L298-N(controllo motori);
- 2 motori DC;
- BreadBoard;
- Batteria 12 Volt;
- 3 poteziometri;
- Elaboratore dell'utente;





05 | **Cenni sul codice**

Codice CONTROLLORE (ESP-32CAM)

INIZIALIZZAZIONE:

Aprire i seguenti canali di comunicazione :

- Socket UDP di comunicazione con MATLAB (su porta 5000).
- Socket UDP di comunicazione con PYTHON (su porta 4210).
- Istanza un canale di comunicazione con il MANTENITORE (ARDUINO) atto a scambiare i campioni tramite protocollo I2C.

Aprire un'interfaccia web che ospiterà lo stream della camera .

ESECUZIONE

- 1) Il campionatore (ESP) provvederà ad acquisire tutti i dati necessari da comunicare al sottosistema di osservazione (MATLAB) e al sottosistema di controllo chiamato anche mantentore (ARDUINO) .
- 2) Dallo script python il campionatore acquisirà solamente le coordinate dell'oggetto quando quest'ultimo verrà inquadrato dalla camera.

Codice PYTHON

Questo script si basa sulla libreria OPENCV con un algoritmo di machine learning pre-addestrato alla ricerca dei colori . Adotta la codifica colore HSV (hue-saturation-value), Tale algoritmo si basa su una soglia di threshold il cui colore viene identificato fra un valore minimo e un valore massimo di colore target .

Codice - Controllore (ARDUINO)

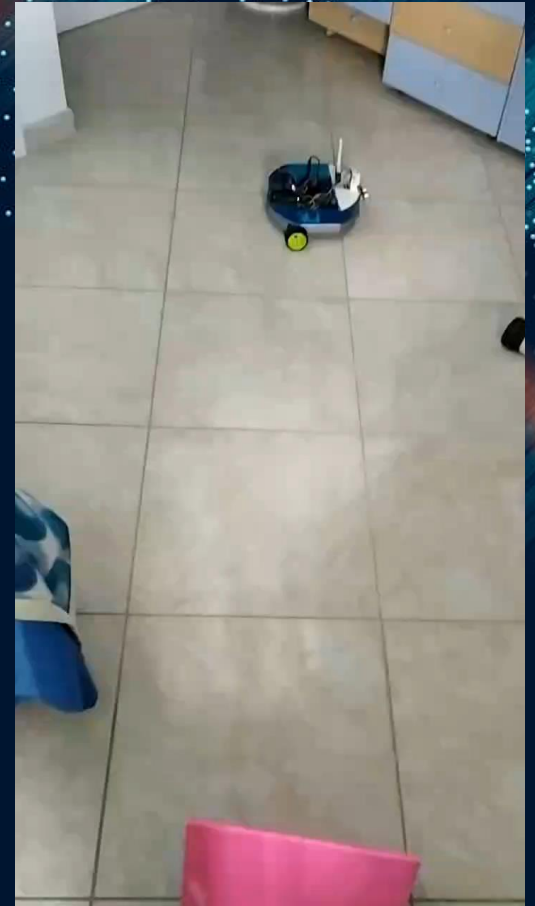
COTROLLORE -> ARDUINO

INIZIALIZZAZIONE:

Apri solamente il canale di ricezione con il campionario (ESP) per ricevere i dati necessari atti all'applicazione della legge di controllo del robot.

ESECUZIONE

- 1) Il controllore permane in una condizione di stasi finché non riceverà il segnale di ENABLE da parte dell'utente che darà input al robot tramite interfaccia MATLAB;
- 2) Una volta abilitato il robot, questo entrerà nella FASE DI RICERCA ;
- 3) individuato il target provvederà ad un suo avvicinamento;
- 4) Si fermerà una volta che sarà arrivato al target ad una distanza impostata dall'utente tramite MATLAB.





06 | **Filtri Digitali**

Filtri Digitali

Il PRIMO filtro viene impiegato per eliminare tutti i byte che contengono informazioni obsolete accumulate nel buffer seriale nella comunicazione ESP32-CAM e ARDUINO (tramite protocollo I2C)

```
byte_serial = Wire.available(); // restituisce il numero di byte accodati nel I2C
trash_byte = byte_serial-pkt_size ;
if(trash_byte>0){
    for(int i = 0 ; i<trash_byte ; i++){ // for che consuma i byte obsoleti su seriale
        Wire.read();
    }
}
```

Il SECONDO filtro è implementato in python e consiste nel creare una finestra di threshold nel individuare un colore target in quanto soggetto a fenomeni di sovraesposizione e sottoesposizione di luce .

```
#impostazione del colore target in base allo spazio colori HUE-SATURATION-VALUE
#hue -> COLORE
# S -> SATURAZIONE
# V -> indica la luminosita del colore

l_h, l_s, l_v = 153, 20,20 # valori minimi
u_h, u_s, u_v = 255, 255, 255 #valori massimi
#la loro differenza indica la soglia di threshold il quale vengono ammessi colori seppur diversi
```

Il TERZO filtro viene impiegato per eliminare disturbi riguardanti le misure .

```
if(k!=0){ // Filtro contro i disturbi di misura
    delay(1000);
    k--;
```



07 | **Ritardi e Campioni**

Problematica della trasmissione del campione e possibile soluzione

Il problema principale che si presenta trattasi che nel momento in cui il robot Deve tornare nella fase di ricerca avrà un ritardo nell'acquisizione dei dati più o meno lungo dovuto a pacchetti accodati nel buffer di rete tra ESP e PYTHON e conseguentemente anche nel buffer d'uscita della seriale tra ESP e ARDUINO. (vedasi video codice PYTHON)



Risoluzione: Occorre che venga eliminata la latenza di fornitura dei dati in caso di un repentino cambiamento di stato del robot. A tal proposito si potrebbe adottare del hardware più performante atto ad eliminare la comunicazione tramite socket UDP tra ESP e PYTHON.

Ritardi e Campionamenti (1)

I campioni vengono creati ad una frequenza che oscilla fra 1-Hz e 29-Hz circa. Tale oscillazione è dovuta a molteplici fattori che combinati, creano una Desincronizzazione tra sotto sistemi , le principali cause attribuibili a tali ritardi sono:

- 1) Hardware low budget(Rallenta pesantemente l'esecuzione del codice in presenza di sottoalimentazioni o saturazione dei buffer di trasmissione e ricezione).
- 2) Saturazione dei buffer dell'interfaccia di rete del router che comunica con MATLAB e PYTHON

Tornando ai nostri campioni, questi verranno trasmessi su i vari sottosistemi tramite un TEMPO DI CAMPIONAMENTO specifico.

ESP32-CAM -> MATLAB tempo trasmissione del campione	1	secondo
ESP32-CAM -> ARDUINO tempo trasmissione del campione	0.5	secondi

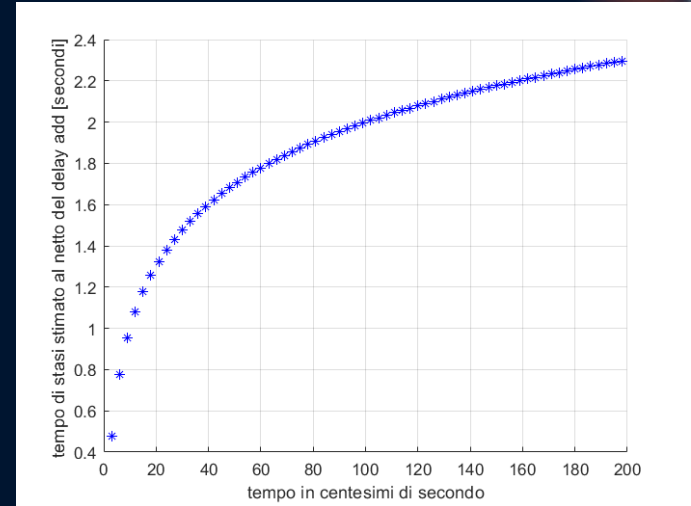
Ritardi e Campionamenti (2)

Per far sì che l'ESP32 generi un campione accettabile, il mantenitore dovrà far permanere il robot in stasi per una quantità di tempo atta a soddisfare il Teorema del campionamento. Per nostra fortuna tale teorema risulta essere soddisfatto se il campionatore ha una frequenza ALMENO doppia rispetto al segnale di ingresso (in questo caso il movimento del robot).

```
delay_sleep = log10(rxData.latency)*1000+delay_add;
```

Istruzione ARDUINO atta a calcolare il tempo di stasi del robot affinché venga rispettato il teorema su citato. Non avendo la possibilità di sapere la latenza del campionatore nel istante in cui effettuerà il campione, verrà considerato la latenza precedente e verrà aggiunto un Delay supplementare configurabile da potenziometro .

(sono inoltre stati inseriti dei potenziometri atti a variare i moltiplicatori di coppia rotazionale e coppia traslazionale dei motori utilissimi a variare la dinamica di movimento del sistema)



08

Socket Matlab

Cenni su
inizializzazione ed
esecuzione del
codice

09

Design del Controllore

Stima parametri del
sistema e design del
controllore

10

Simulazioni

Simulink

11

Sistema Reale

Video dimostrativi



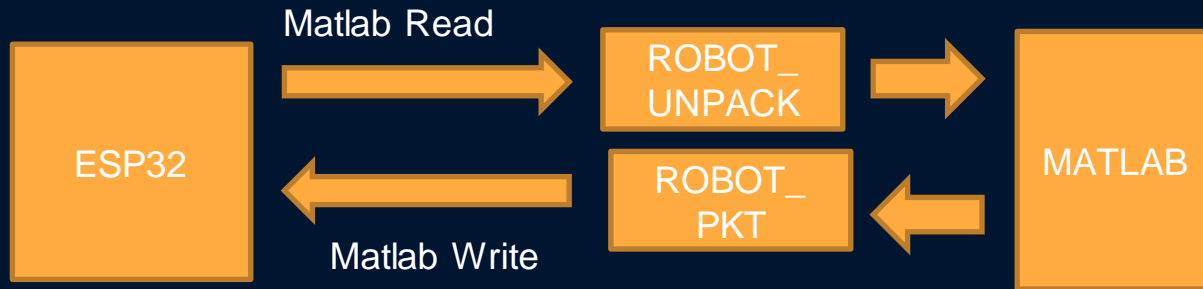
08

Socket Matlab

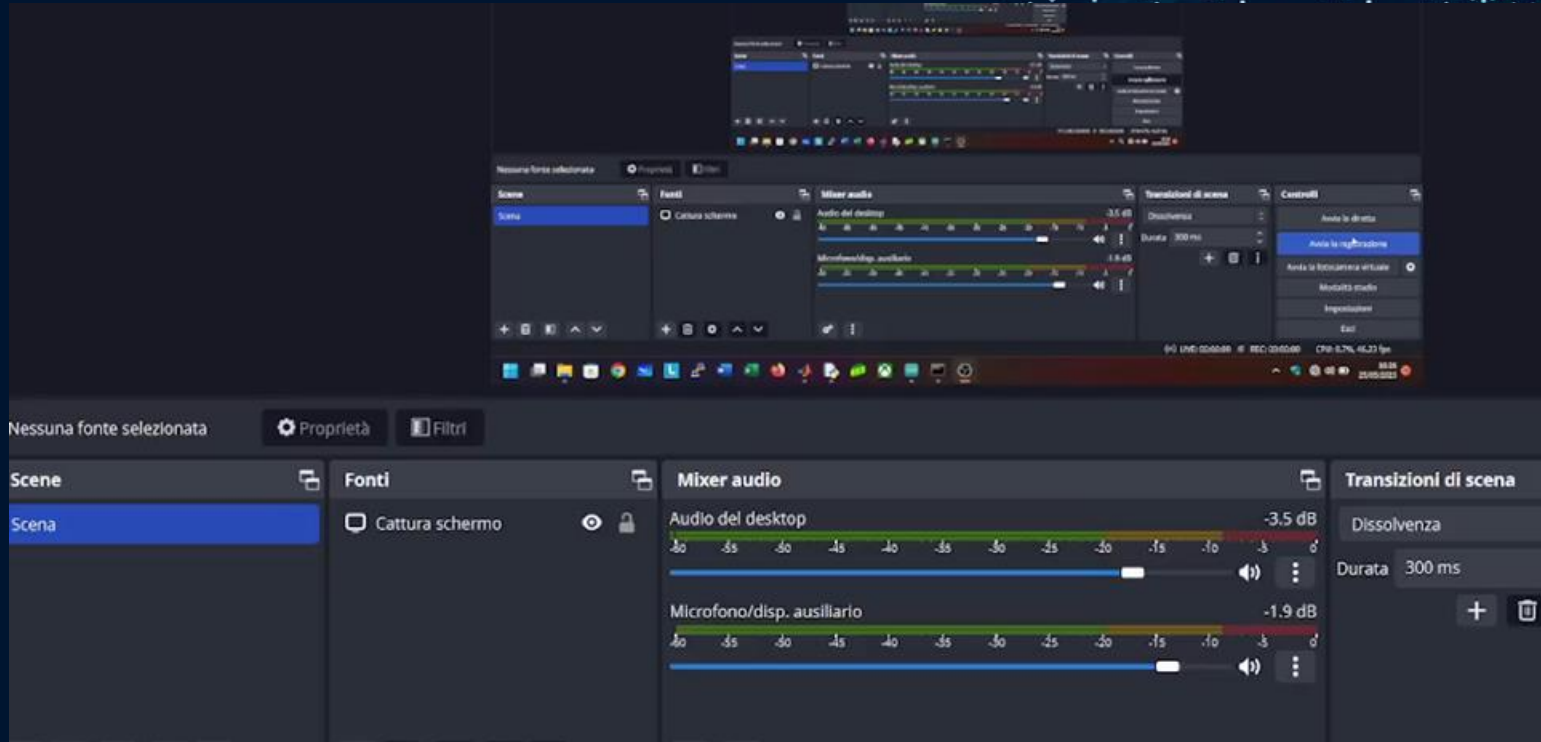
Socket Matlab

```
19
20     localIP = '192.168.1.3'; % indirizzo ip del PC
21     localPort = 5006;
22     udp_server = udpport('LocalHost', localIP, 'LocalPort', localPort);
23     remoteIP = '192.168.1.6'; % indirizzo ip del robot
24     remotePort = 5000;
25     %% Comandi Iniziali
26     % valori cambiati dall utente (mandato all ESP)
27     dataOut.last_cmd = 1; % OK 0 = stop , 1 = start ,
28     dataOut.prefer_distance = 60; % distanza custom di inseguimento dell
29
30     % communication
```

Parte iniziale di codice socket UDP in Matlab «Main Sender»



Video Dimostrativo (2) socket MATLAB



Dati: 1. Enable 2.d_ref 3. Latency 4.pos_x 5.pos_y 6.distance



09

Design del Controllore

Design del Controllore

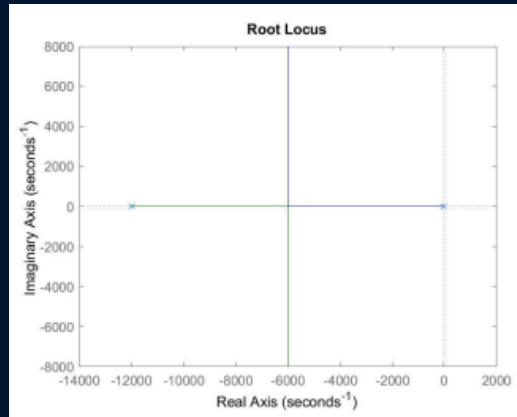
```
%Parametri Progetto
%Dimensioni
r=0.04; %[m] %raggio ruote
w=0.02; %[m] larghezza ruote
m=0.1; % [Kg] massa ruota
J=0.5*m*(r^2+(w^2)/6);%kg*m^2 momento
% di inerzia ruote
d=0.2; %[m] %Interasse
Km=100; %[N*m/A] Proporzione tra
% velocità angolare e corrente applicata.
c=1; %N*m*s

%Parametri per simulazione simulink
omegaR=0; %Condizioni iniziali
omegaL=0; % Condizioni iniziali
dmin=0.1; %distanza rif
thetamin=0.2; %angolo rif
Kv1=1;%costanti per legge
% proporzionale delle velocità
Kv2=1;
fs=29; %Frequenza di campionamento
Ts=1/fs; %T di campionamento
```

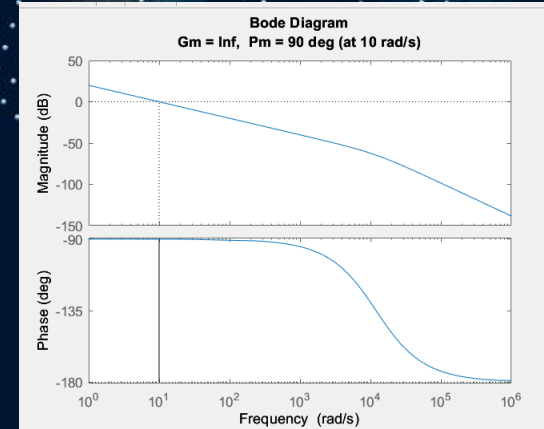
Parametri uniciclo

```
%% Simulazione a tempo continuo
%Sicliche: Errore nullo per riferimento costante.
s=tf('s');
P1=Km*alpha/(s*(J*s+c));
figure(1)
rlocus(P1);
K1=0.1;
C1=K1;
L1=minreal(C1*P1);
figure(2)
rlocus(L1);
figure(3)
nyquist(L1)
%I poli del sistema a ciclo chiuso
% si trovano nel semipiano sinistro
figure(4)
[Gm,Pm,Wcg,Wcp]=margin(L1);
margin(L1);
ritardomax=Pm*(pi/180)/Wcp;
Wrl=minreal(L1/(1+L1));
figure(5);
step(Wrl);
```

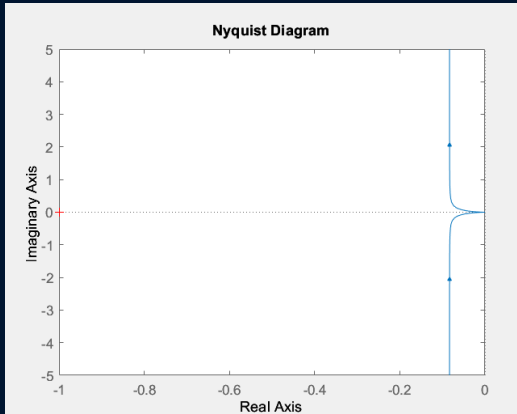
Design controllore



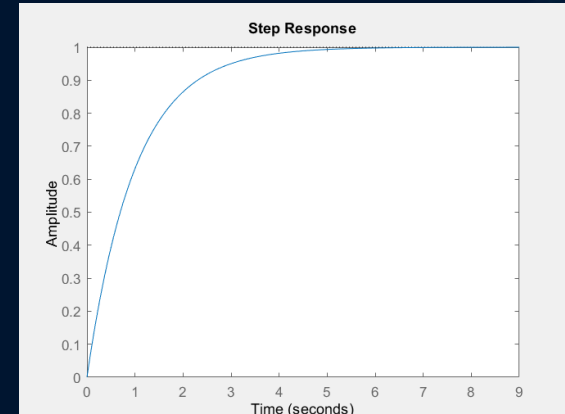
Root Locus



Bode



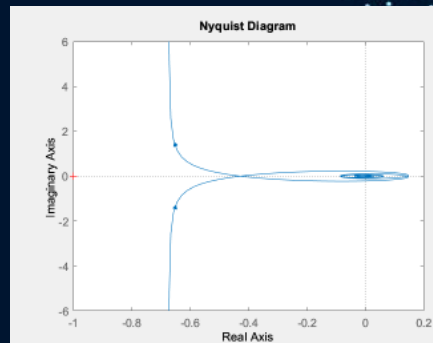
Nyquist



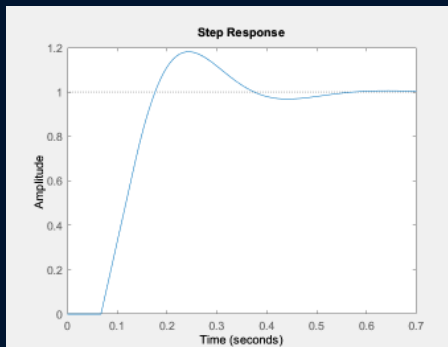
Risposta al gradino

Controllore digitale e ritardi

```
%% Controllore digitale
%simuliamo con approssimazione di Padè
%gli effetti del controllore digitale e dei
% ritardi di comunicazione
rit=exp(-(Ts/2)+0.05)*s);
Lr=minreal(rit*L1);
figure(6)
nyquist(Lr)
Wrr=minreal(Lr/(1+Lr));
figure(7);
step(Wrr);
```



Nyquist

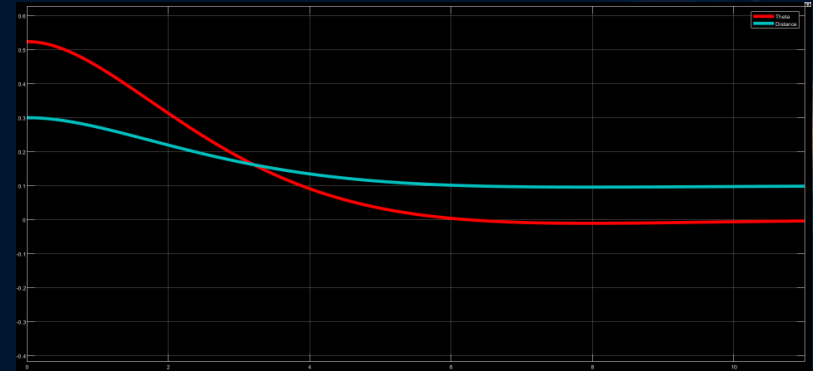
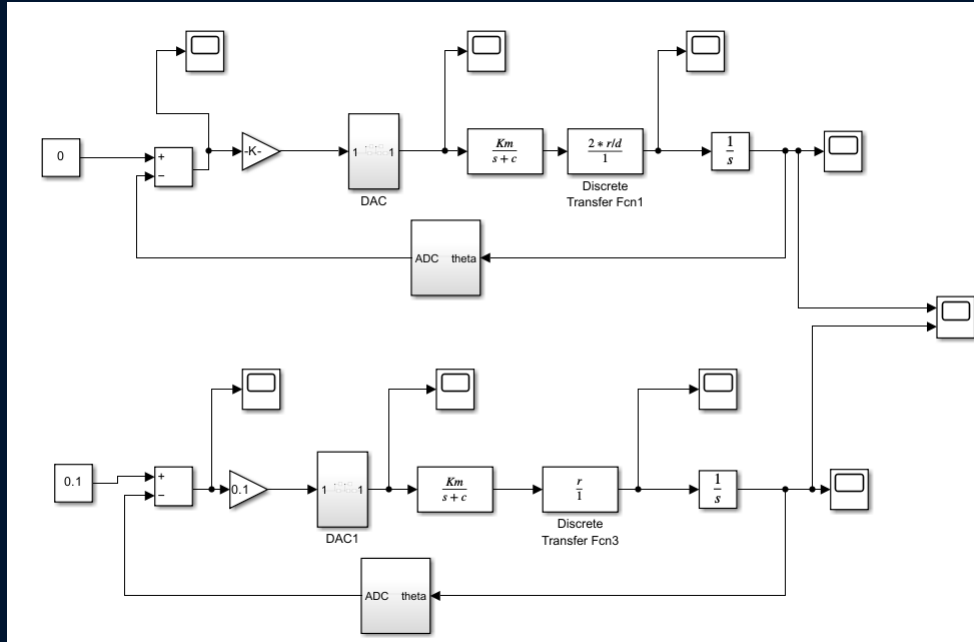


Risposta al gradino sistema con ritardo

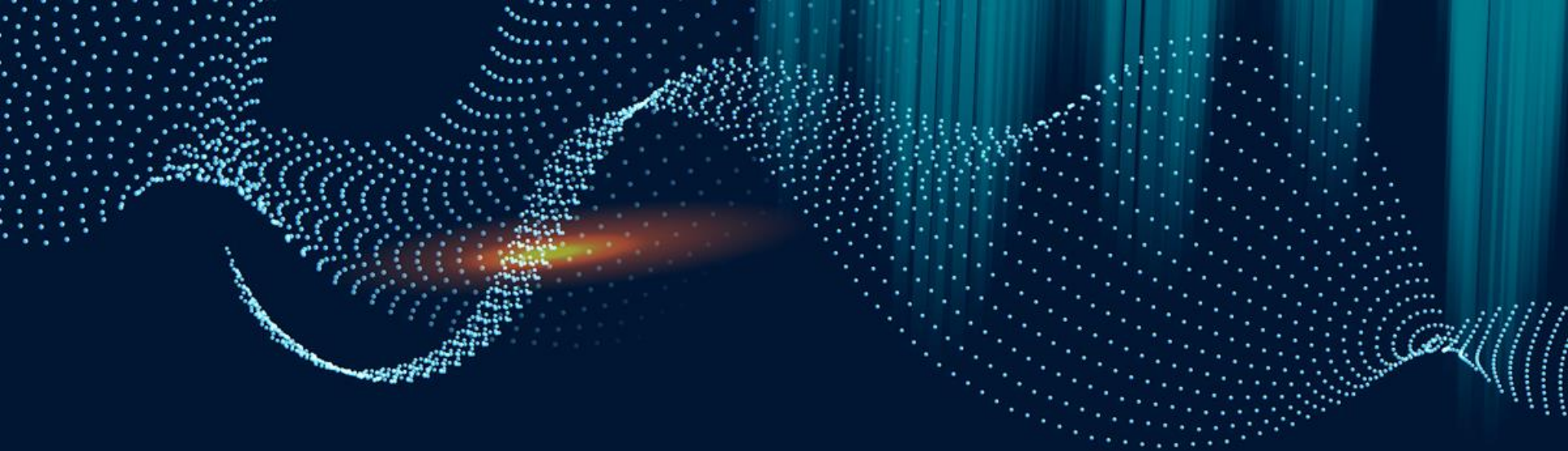


10 | **Simulink**

Simulink



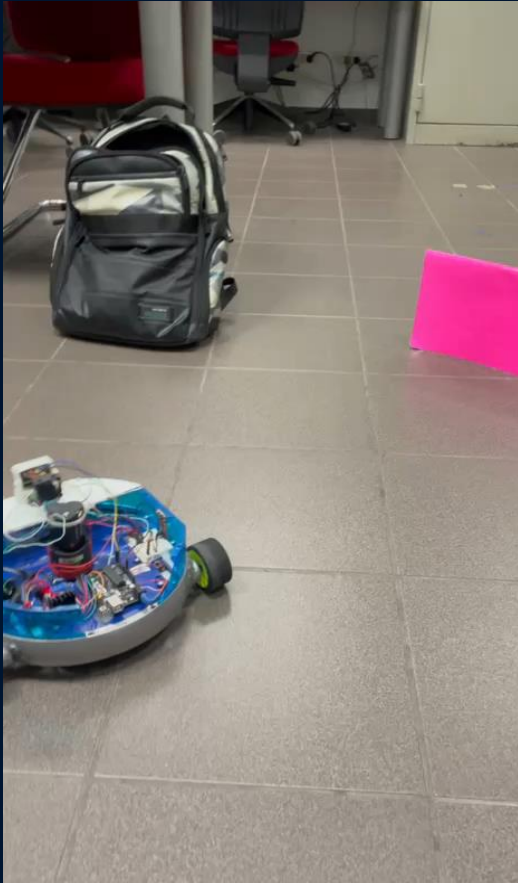
Blu: Andamento della distanza. C.I. 30 cm
Rosso: Andamento theta C.I. 30 deg



11

Sistema Reale

Video Dimostrativi (3) ROBOT



Funzionamento ROBOT (1)



Funzionamento ROBOT (2)

AZIONI SVOLTE DAL
ROBOT:

- 1) RICERCA
- 2) AVVICINAMENTO