Progettazione e Sviluppo Sperimentale di Sistemi di Controllo su Piattaforme Embedded

Università degli Studi **Mediterranea** di Reggio Calabria

Relatore
Prof. Valerio Scordamaglia

Candidato
Salvatore Avellino

Correlatore
Vito Antonio Nardi

INTRODUZIONE

- Negli ultimi decenni, si è assistito ad un aumento esponenziale di applicazioni in cui l'utilizzo di sistemi elettronici è diventato una costante da cui non è più possibile prescindere: tali apparati elettronici sono sistemi computer-based normalmente detti *embedded systems*.
- L'ampio ventaglio di possibili applicazioni sta mantenendo alta l'attenzione su tali sistemi, e a giudicare dalle previsioni di crescita economica del mercato embedded, tale interesse non è destinato a diminuire.
- Il presente lavoro di tesi mira a proporre degli esempi di utilizzo della tecnologia embedded nell'ambito dei sistemi di controllo, affrontando il problema dello sviluppo e dell'implementazione di alcuni algoritmi di controllo digitale per due setup sperimentali.
- I sistemi di controllo sono stati implementati sulle piattaforme embedded Arduino MEGA 2560, STM32 NUCLEO F401RE e Raspberry Pi 3 model B.
- Al fine di verificare la bontà del processo di sviluppo, la validazione delle prestazioni degli algoritmi di controllo è avvenuta confrontando i risultati sperimentali con quelli simulativi.

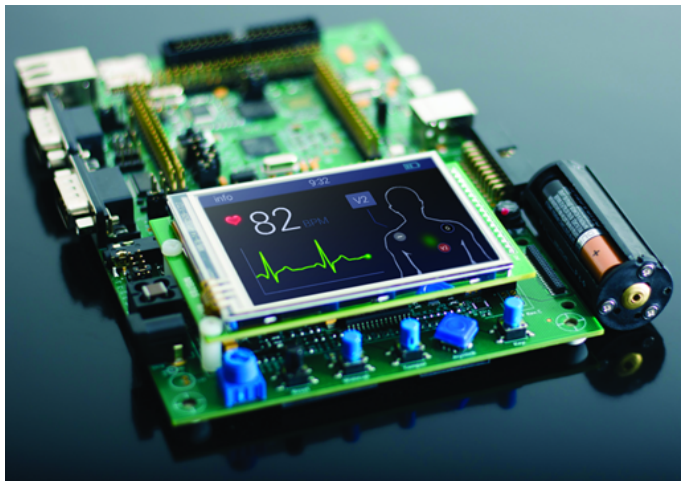
Sistemi Embedded

Per sistema embedded si tende ad indicare l'insieme composto da hardware e software dedicato a specifici scopi, i cui elementi siano tutti quanti integrati ed incorporati[1].

- Bassi consumi di energia
- Adattabilità ed affidabilità
- Dimensioni e costi contenuti

Capacità di interazione col mondo esterno.

- Periferiche di comunicazione seriale
- GPIO, moduli PWM, ADC, ecc.



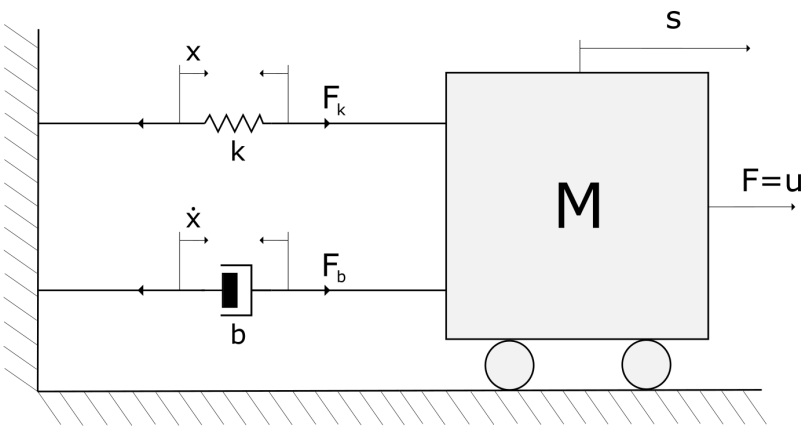
Applicazioni

Trasporti, applicazioni elettriche ed elettroniche, controllo dei processi, telecomunicazioni, sicurezza, ecc.

[1] Carraturo Alexja, Trentini Andrea - Sistemi Embedded: Teoria e Pratica. Ledizioni (2017).

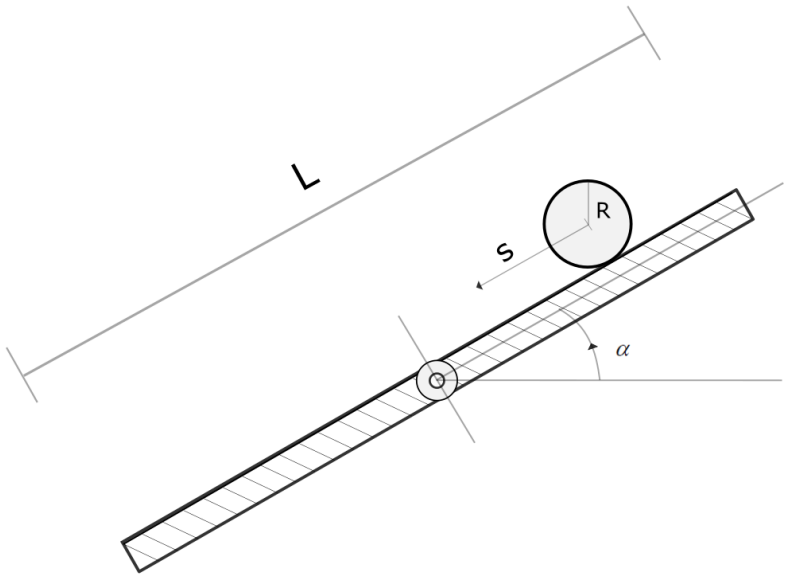
Sistemi di Interesse

Massa-Molla-Smorzatore (stabile)



$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = -\frac{k}{M}x_1 - \frac{b}{M}x_2 + \frac{1}{M}u \\ y = x_1 \end{cases}$$

Ball-Beam (instabile)



$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = \frac{5}{7}gu \\ y = x_1 \end{cases}$$

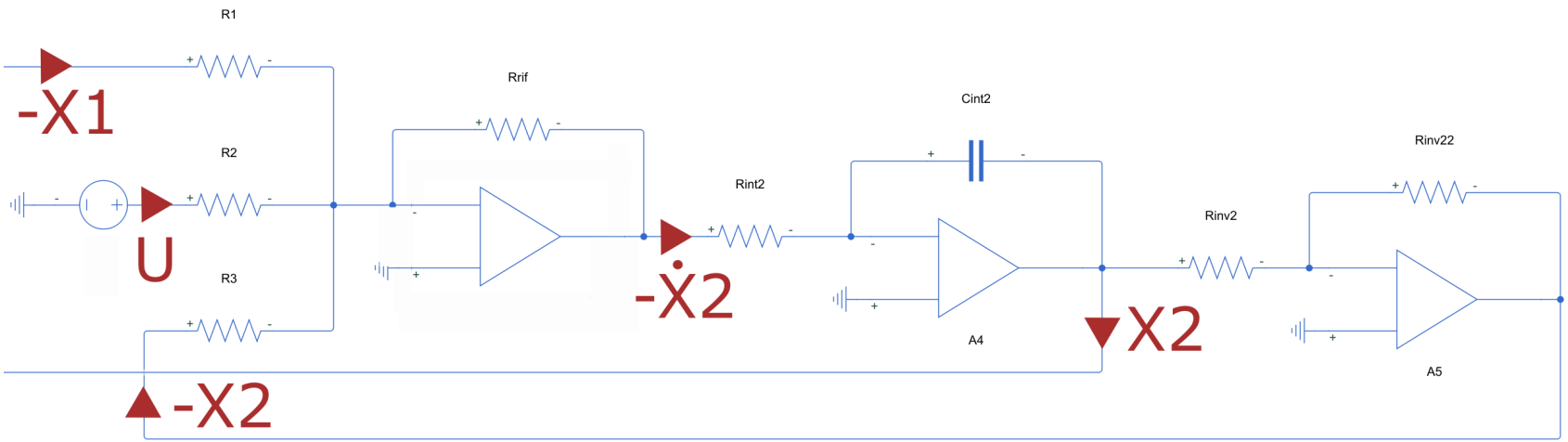
Traduzione Modello Matematico

Esempio Massa-Molla-Smorzatore - Stato x_2

$$\dot{x_2} = -\frac{k}{M}x_1 - \frac{b}{M}x_2 + \frac{1}{M}u$$

$$\Updownarrow$$

Sommatore Invertente | Integratore Invertente | Amplificatore Invertente



Traduzione Modello Matematico

Relazioni tra i componenti passivi delle configurazioni.

SOMMATORE INV. $\Longrightarrow$ $\frac{R_{rif}}{R_1} = \frac{k}{M}$ - $\frac{R_{rif}}{R_2} = \frac{1}{M}$ - $\frac{R_{rif}}{R_3} = \frac{b}{M}$

INTEGRATORE INV. $\Longrightarrow$ $R_{int2} C_{int2} = 1$

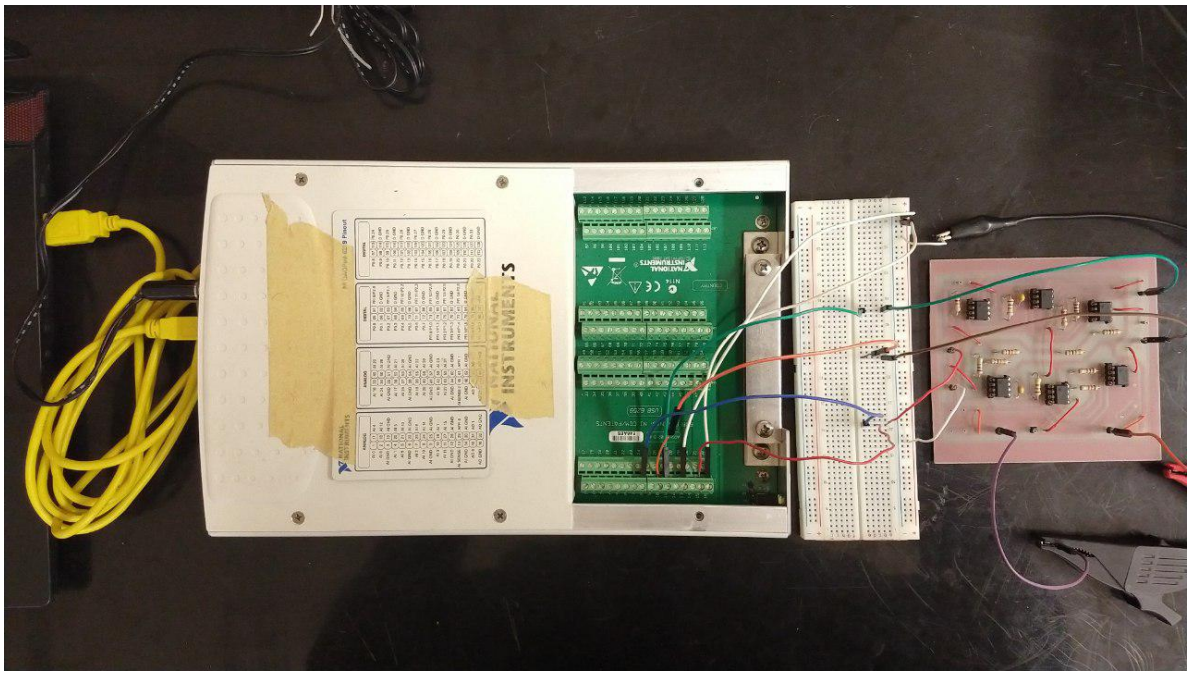
AMPLIFICATORE INV. $\Longrightarrow$ $R_{inv2} = R_{inv22}$

Tra le infinite combinazioni di elementi passivi compatibili con i vincoli, è stata scelta la configurazione che minimizza il consumo di corrente.

VERIFICA COMPORTAMENTO

Valutazione delle dinamiche a ciclo aperto dei circuiti di prova, per mezzo di una scheda DAQ NI 6259.

- Generazione ingresso
- Acquisizione segnali
- Importazione su MATLAB
- Confronto risultati sperimentali e simulativi



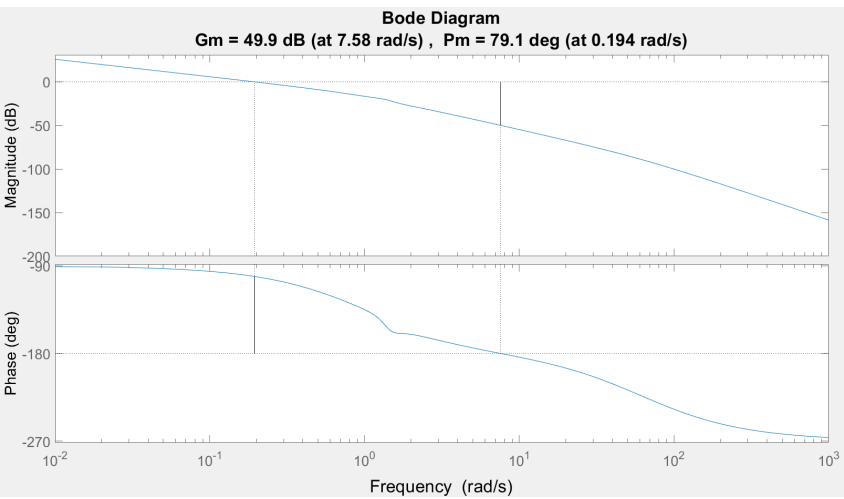
Specifiche e Prestazioni

Massa-Molla-Smorzatore

- Errore a regime nullo per riferimenti di tipo gradino
- Sovra-elongazione massima del 10%
- Tempo di assestamento al 3% di 30 s

$\Updownarrow$

- $\omega_c = 0.1977\ rad/s$
- $M_f \simeq 82.4834°$



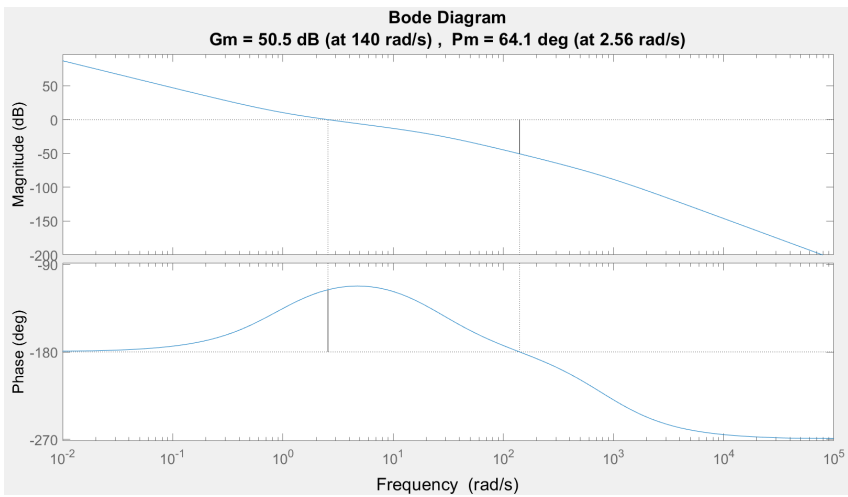
Funzione di anello per sistema M-M-S

Ball-Beam

- Errore a regime nullo per riferimenti di tipo gradino
- Sovra-elongazione massima del 20%
- Tempo di assestamento al 3% di 3 s

$\Updownarrow$

- $\omega_c = 2.5636\ rad/s$
- $M_f \simeq 64.2561°$



Funzione di anello per sistema B-B

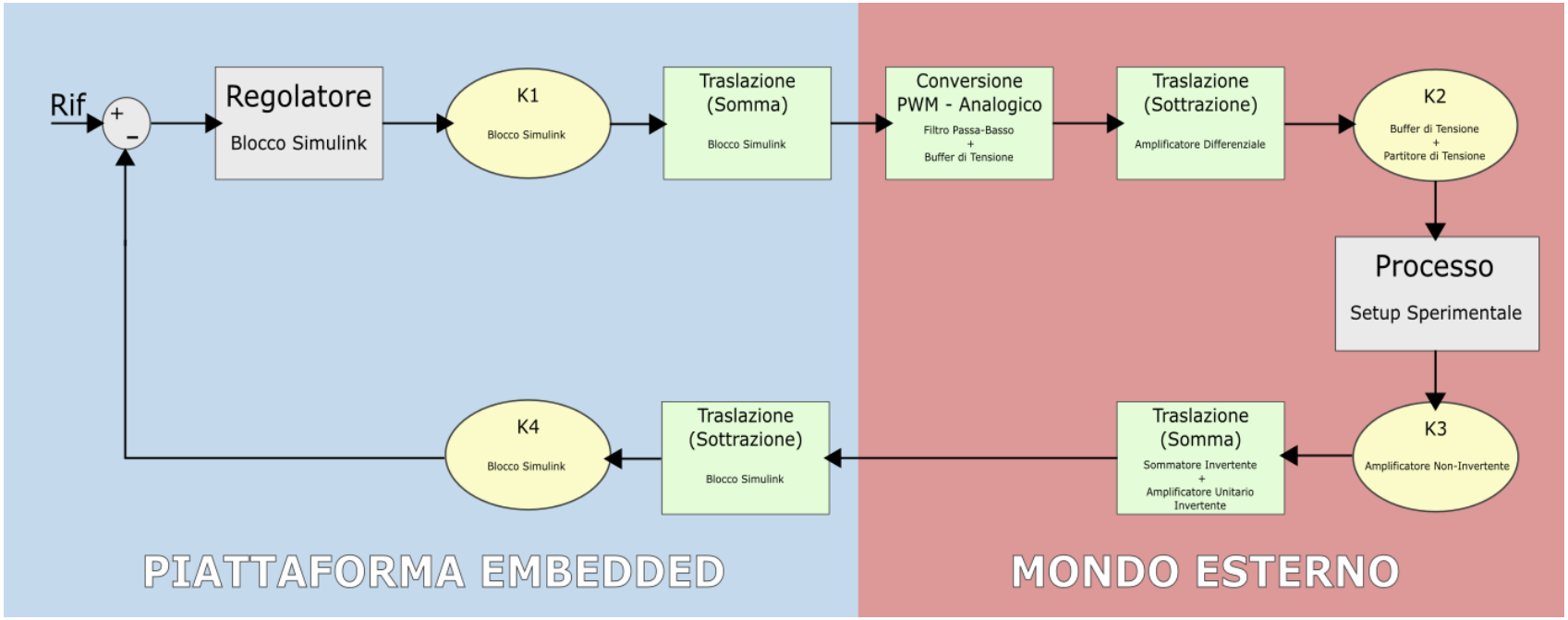
Confronto Piattaforme Embedded

	Arduino MEGA 2560	**STM32 NUCLEO F401RE**	**Raspberry Pi 3 model B**
CLK_{cpu}	16 Mhz	84 Mhz	1.2 Ghz
OS	single-task	multi-task RT	multi-task
V_{range}	0 V $\rightarrow$ 5 V	0 V $\rightarrow$ 3.3 V	0 V $\rightarrow$ 3.3 V
ADC	presente	presente	assente
RIS_{adc}	10 bit $\Rightarrow$ 4.88 mV	12 bit $\Rightarrow$ 0.805 mV	-
F_{pwm}	480 Hz - 980 Hz	$\leq$ 12 MHz	$\leq$ 19.2 MHz

Frequenza e risoluzione utilizzate per il segnale PWM.

	Arduino MEGA 2560	**STM32 NUCLEO F401RE**	**Raspberry Pi 3 model B**
F_{pwm}	980 Hz	980 Hz	1171.875 Hz
RIS_{pwm}	8 bit $\Rightarrow$ 19.5 mV	8 bit $\Rightarrow$ 12.9 mV	10 bit $\Rightarrow$ 3.22 mV

SISTEMA DI CONTROLLO

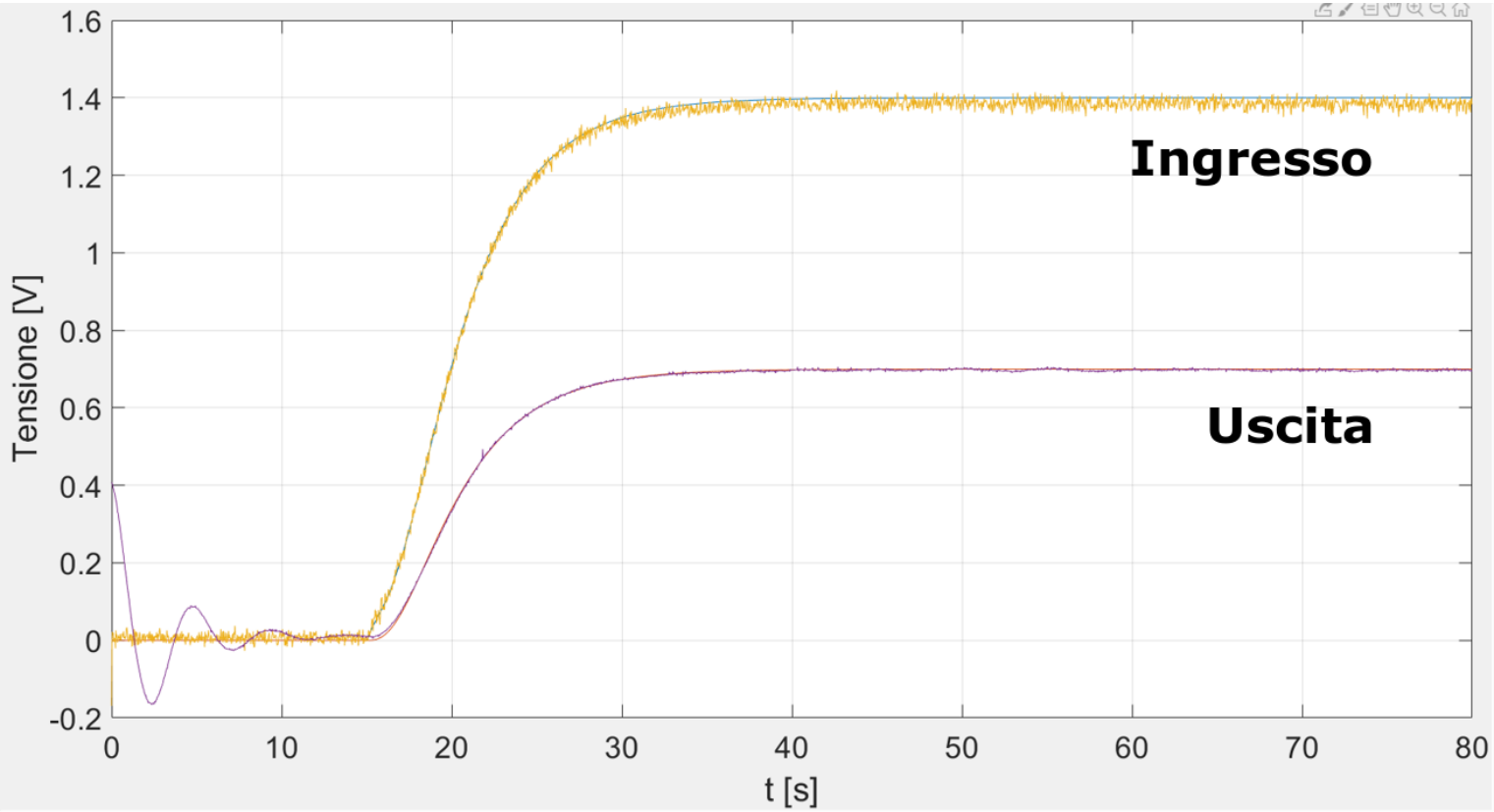


SCHEMA DI RETROAZIONE UNITARIA

PIATTAFORMA EMBEDDED	$\Longrightarrow$	Confronta l'uscita del processo col riferimento e calcola l'azione di controllo.
SETUP SPERIMENTALE	$\Longrightarrow$	Rappresenta il processo a cui imporre il comportamento desiderato.
INTERFACCIA	$\Longrightarrow$	Circuiti e blocchi Simulink per gestire le tensioni.

STM32 NUCLEO F401RE - MASSA-MOLLA-SMORZATORE

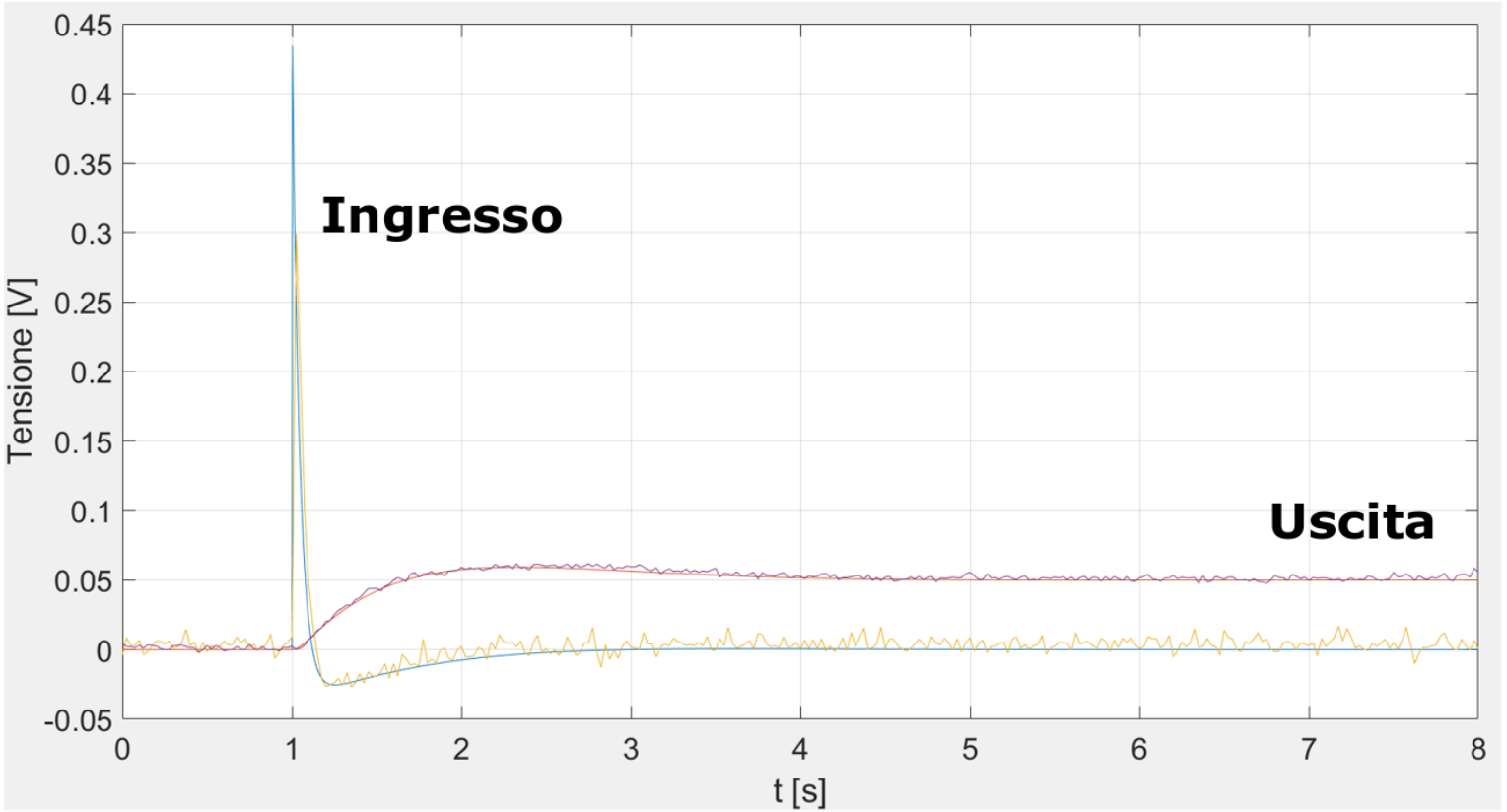
Riferimento Gradino $\Longrightarrow$ Istante applicazione = 15 s - Ampiezza gradino = 0.7



Ingresso/Uscita **Simulati** $\rightarrow$ BLU / ROSSO Ingresso/Uscita **Acquisiti** $\rightarrow$ GIALLO / VIOLA

ARDUINO MEGA 2560 - BALL-BEAM

Riferimento Gradino $\Longrightarrow$ Istante applicazione = 1 s - Ampiezza gradino = 0.05



Ingresso/Uscita **Simulati** $\rightarrow$ BLU / ROSSO Ingresso/Uscita **Acquisiti** $\rightarrow$ GIALLO / VIOLA

CRITICITÀ IMPLEMENTAZIONE SU RASPBERRY

Sistema operativo multi-task non real-time $\Longrightarrow$ *Installazione patch RT*

- Rende il kernel Linux fully-preemptible, permettendo l'esecuzione di processi real-time.

ADC integrato assente $\Longrightarrow$ *Impiego di ADC esterno ADS1115*

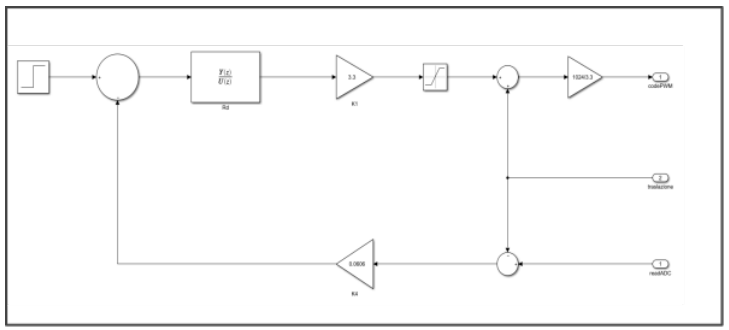
- Permette l'acquisizione di segnali analogici ma non rientra nell'hardware supportato da Simulink.

Generazione PWM hardware preferibile a quella software

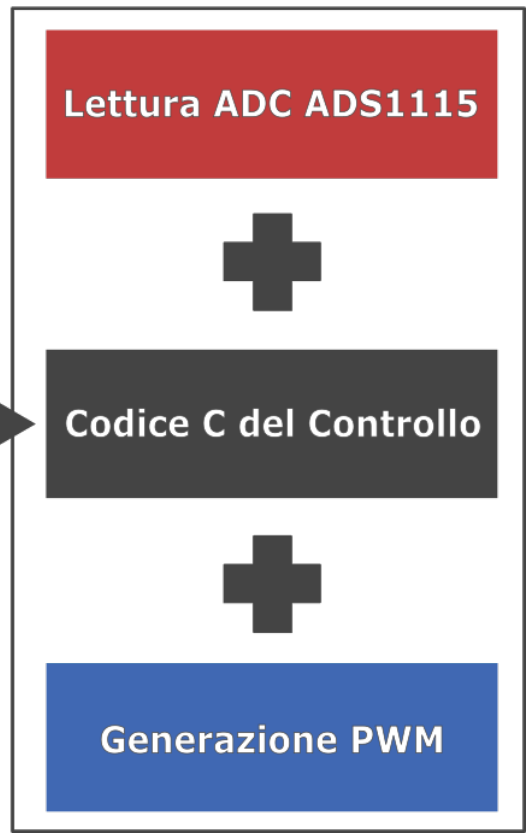
- Per generare PWM hardware si ricorre ad una libreria esterna non presente in Simulink.

CODICE AUTOGENERATO

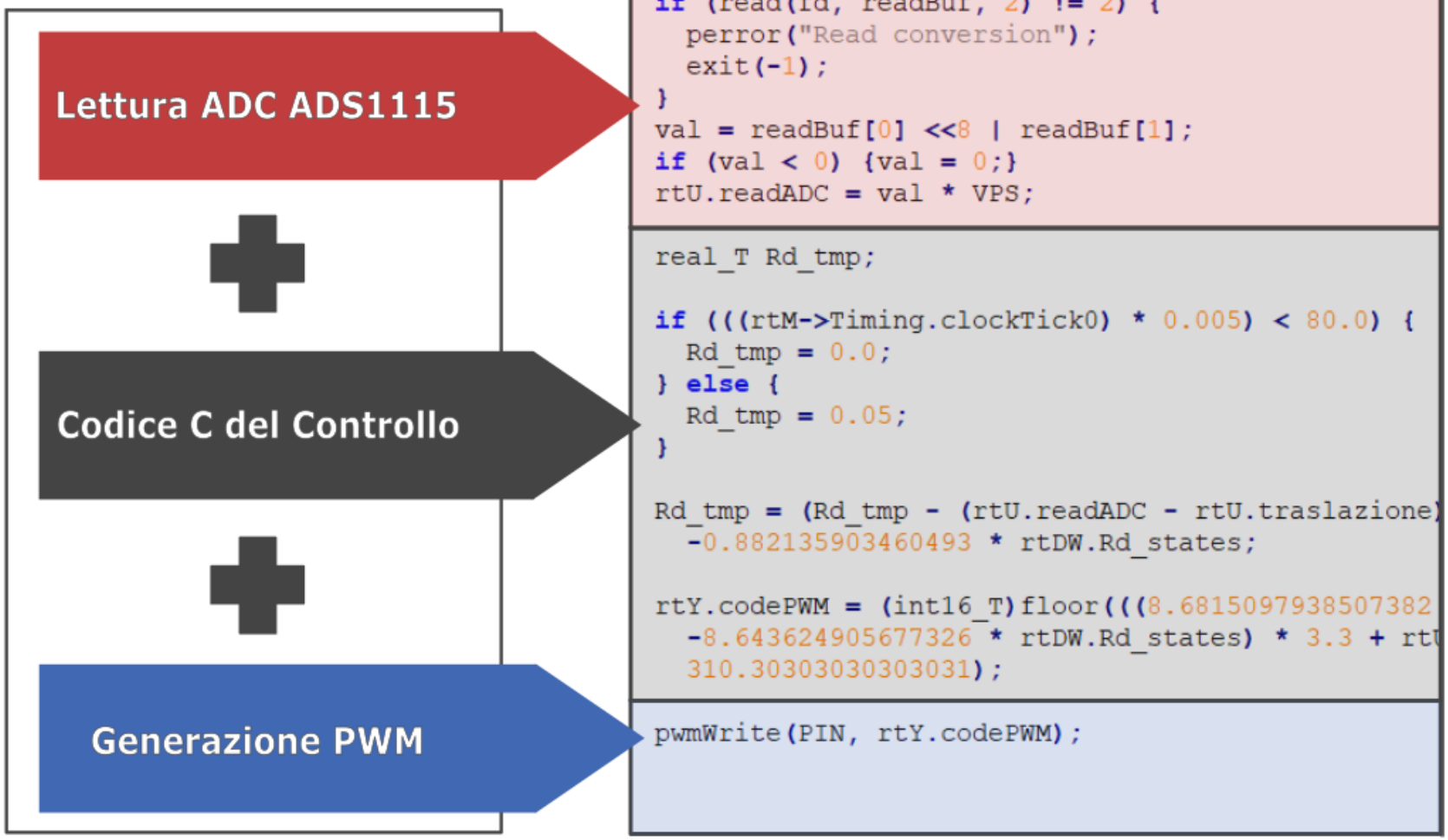
Si ricorre ai tool *Code Generation Advisor* ed *Embedded Coder* dell'ambiente di sviluppo MATLAB/Simulink, per generare automaticamente del codice C da modificare manualmente per l'applicazione in oggetto.



```
void controlloPWM_step(void)
{
  real_T rtb_Sum1;
  real_T rtb_Switch;
  real_T Step;

  /* Step: '<Root>/Step' */
  if (((rtM->Timing.clockTick0) * 0.031777629784490134) < 10.0) {
    Step = 0.0;

    /* Switch: '<Root>/Switch' incorporates:
     *  Constant: '<Root>/costNulla'
     */
    rtb_Switch = 0.0;
  } else {
    Step = 0.5;

    /* Switch: '<Root>/Switch' incorporates:
     *  DiscreteTransferFcn: '<Root>/Fd'
     *  Inport: '<Root>/readADC'
     *  Inport: '<Root>/traslazione'
     *  Sum: '<Root>/Sum'
     *  Sum: '<Root>/Sum2'
     *
     * Block description for '<Root>/readADC':
     *  Valore analogico letto dall'adc sul canale A0!
     *
     * Block description for '<Root>/traslazione':
     *  Valore analogico letto dall'adc sul canale A2!
     */
    rtb_Switch = 0.031278026956042683 * rtDW.Fd_states - (rtU.readADC -
      rtU.traslazione);
  }

  /* End of Step: '<Root>/Step' */
```

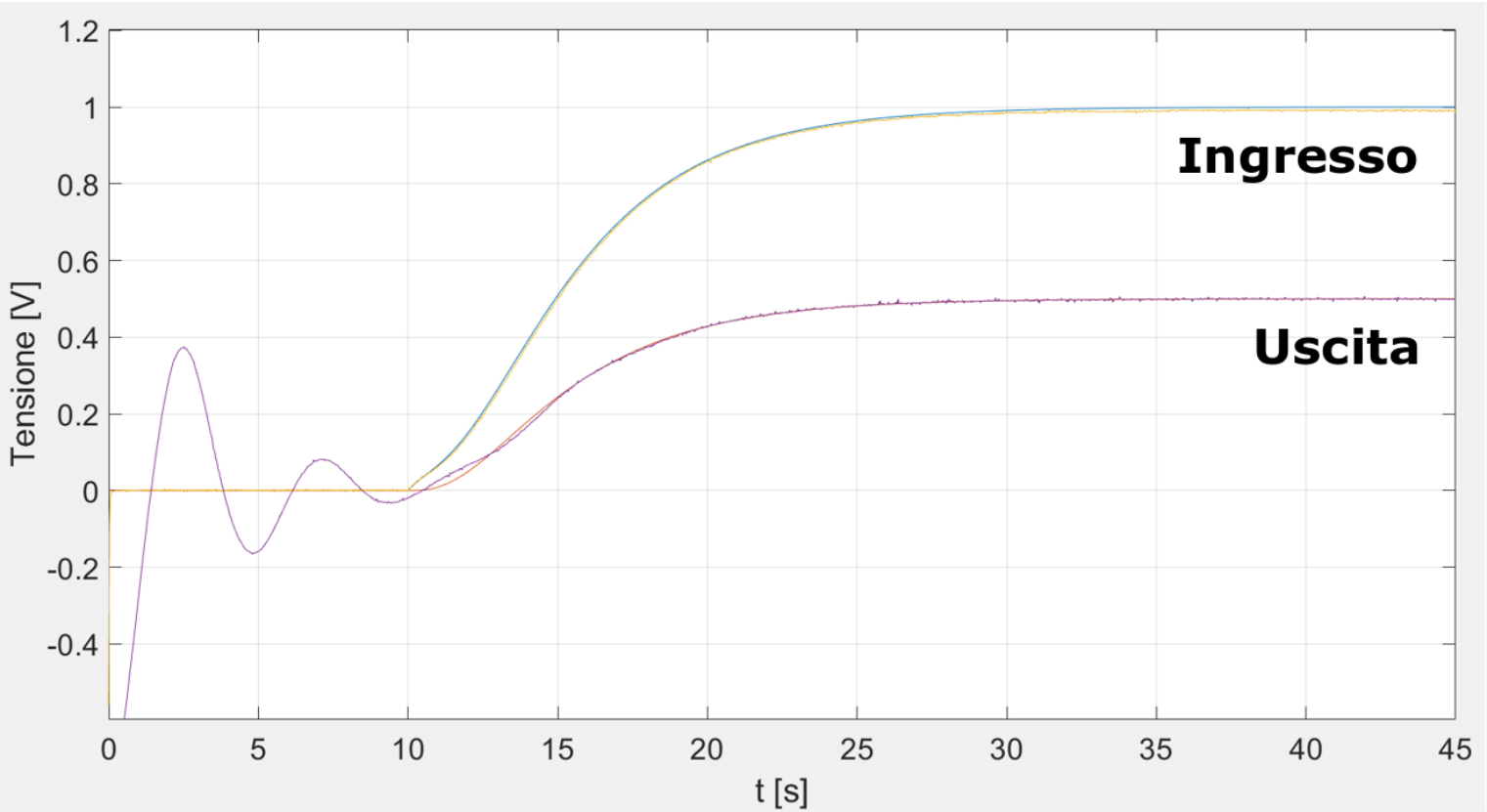


Codice Autogenerato



RASPBERRY PI 3 MODEL B - MASSA-MOLLA-SMORZATORE

Riferimento Gradino $\Longrightarrow$ Istante applicazione = 10 s - Ampiezza gradino = 0.5



Ingresso/Uscita **Simulati** $\rightarrow$ BLU / ROSSO Ingresso/Uscita **Acquisiti** $\rightarrow$ GIALLO / VIOLA

Verifica Requisiti Temporali

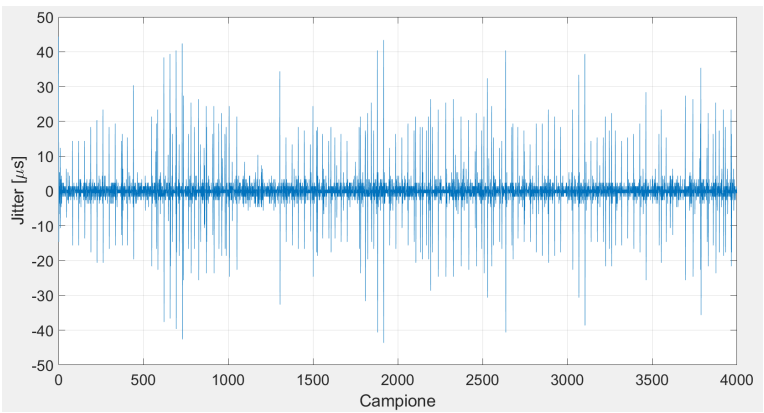
Jitter

Errore temporale commesso ad ogni iterazione del processo, per un dato tempo di esecuzione (*deadline*).

Massa-Molla-Smorzatore

Valori massimi di jitter ed incertezza sul tempo di campionamento:

- $T_j = 146\ \mu s$
- $\Delta = 0.24\%$

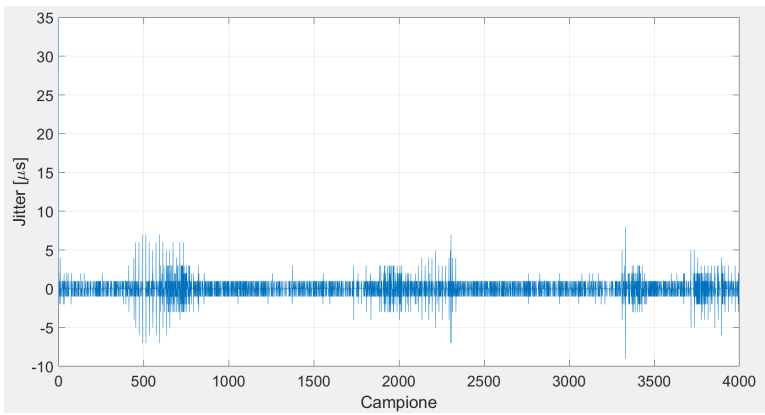


Errore relativo ad istante di avvio della routine per M-M-S

Ball-Beam

Valori massimi di jitter ed incertezza sul tempo di campionamento:

- $T_j = 182\ \mu s$
- $\Delta = 1.78\%$



Errore relativo ad istante di avvio della routine per B-B

Conclusioni

- Sono stati realizzati due circuiti analogici i cui comportamenti riproducono le dinamiche del sistema Massa-Molla-Smorzatore e del sistema Ball-Beam.
- Sono stati progettati due algoritmi di controllo digitale per la regolazione dei setup sperimentali, implementati utilizzando i tool di generazione automatica del codice di controllo presenti nell'ambiente di sviluppo MATLAB/Simulink (*Code Generation Advisor* ed *Embedded Coder*).
- I regolatori sono stati realizzati sulle piattaforme embedded Arduino MEGA 2560, STM32 NUCLEO F401RE e Raspberry Pi 3 model B.
- Il sistema operativo del Raspberry è stato predisposto all'esecuzione di processi real-time, ed è stata utilizzata una procedura *ad hoc* per l'interfacciamento del codice di controllo autogenerato con le periferiche di input/output della piattaforma embedded.
- Per tutti i setup sperimentali, la validazione delle prestazioni degli algoritmi di controllo è avvenuta con successo, confrontando i risultati ottenuti sperimentalmente con quelli ottenuti per via simulativa.

Progettazione e Sviluppo Sperimentale di Sistemi di Controllo su Piattaforme Embedded

Università degli Studi **Mediterranea** di Reggio Calabria

Relatore
Prof. Valerio Scordamaglia

Candidato
Salvatore Avellino

Correlatore
Vito Antonio Nardi

Vincoli del problema di ottimizzazione

$$\frac{1}{R_{int1}}\dot{x_1} + \frac{1}{R_{inv1}}x_1 + \frac{k}{MR_{rif}}x_1 \leq I_{max}$$

$$\frac{1}{R_{inv1}}x_1 \leq I_{max}$$

$$\frac{1}{R_{int2}}\dot{x_2} + \frac{1}{R_{rif}}\dot{x_2} \leq I_{max}$$

$$\frac{1}{R_{int1}}x_2 + \frac{1}{R_{int2}}\dot{x_2} + \frac{1}{R_{inv2}}x_2 \leq I_{max}$$

$$\frac{1}{R_{inv2}}x_2 + \frac{b}{MR_{rif}}x_2 \leq I_{max}$$

Indice di costo

$$F = \Delta I_1^2 + \Delta I_2^2 + \Delta I_3^2 + \Delta I_4^2 + \Delta I_5^2$$

- Ogni delta rappresenta la differenza tra un valore di corrente ideale liberamente definibile dall'utente (purché inferiore ad I_{max}) e le i-esime correnti effettivamente erogate dagli amplificatori in base ai vari resistori e condensatori utilizzabili.

Argomenti della funzione *fmincon*: i vincoli sono espressi nella forma $Ax \leq b$.

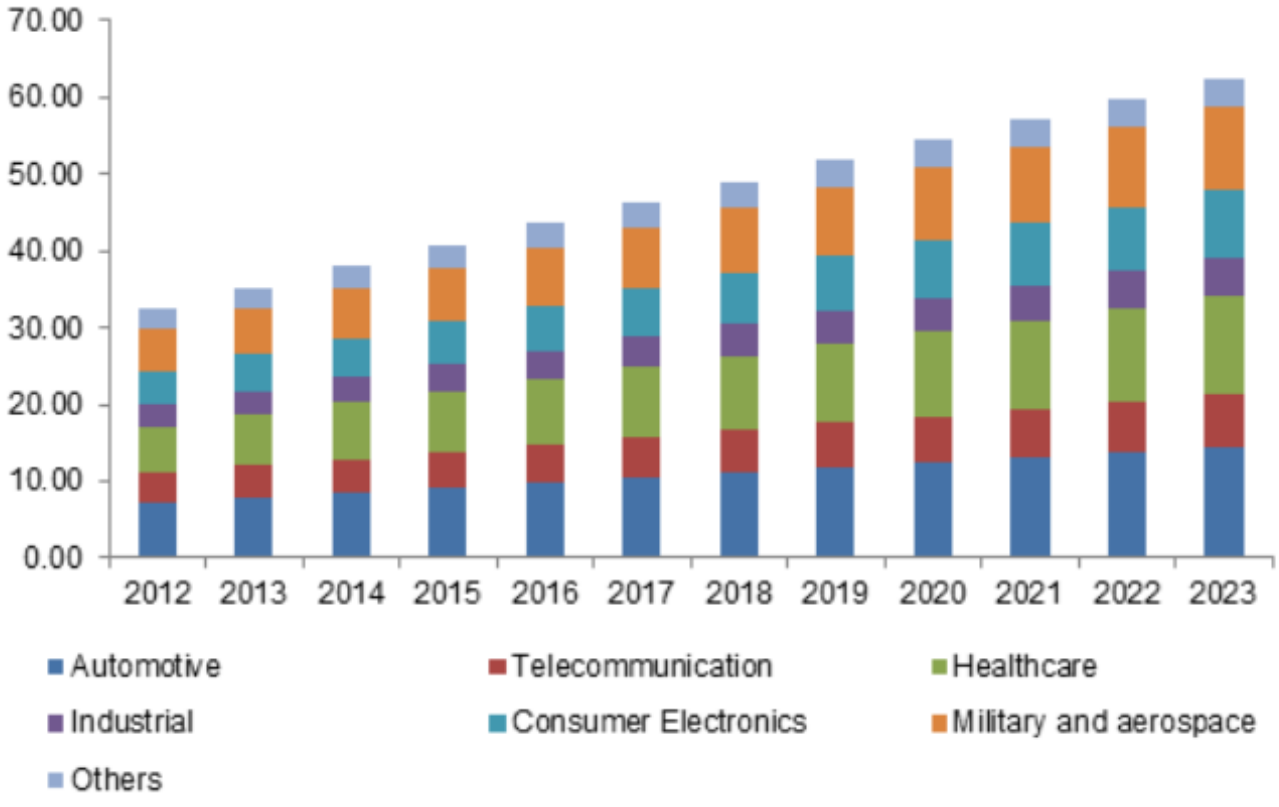
$$b = \begin{bmatrix} I_{max} \\ I_{max} \\ I_{max} \\ I_{max} \\ I_{max} \end{bmatrix}$$

$$x = G = \begin{bmatrix} G_{int1} \\ G_{int2} \\ G_{inv1} \\ G_{inv2} \\ G_{rif} \end{bmatrix}$$

$$A = V = \begin{bmatrix} \dot{x_1} & 0 & x_1 & 0 & \frac{k}{M} x_1 \\ 0 & 0 & x_1 & 0 & 0 \\ 0 & \dot{x_2} & 0 & 0 & \dot{x_2} \\ x_2 & \dot{x_2} & 0 & x_2 & 0 \\ 0 & 0 & 0 & x_2 & \frac{b}{M} x_2 \end{bmatrix}$$

Stime Mercato Embedded

Nel 2015 il mercato globale dei sistemi embedded era stato valutato essere di 159.00 miliardi di dollari americani, ed era stato stimato che per la fine del 2021 avrebbe raggiunto un valore di 225.34 miliardi; ricerche di mercato più recenti, confermano il trend di crescita arrivando a dichiarare che supererà i 258.72 miliardi nel 2023.



Dimensione mercato europeo (in miliardi di dollari) per sistemi embedded

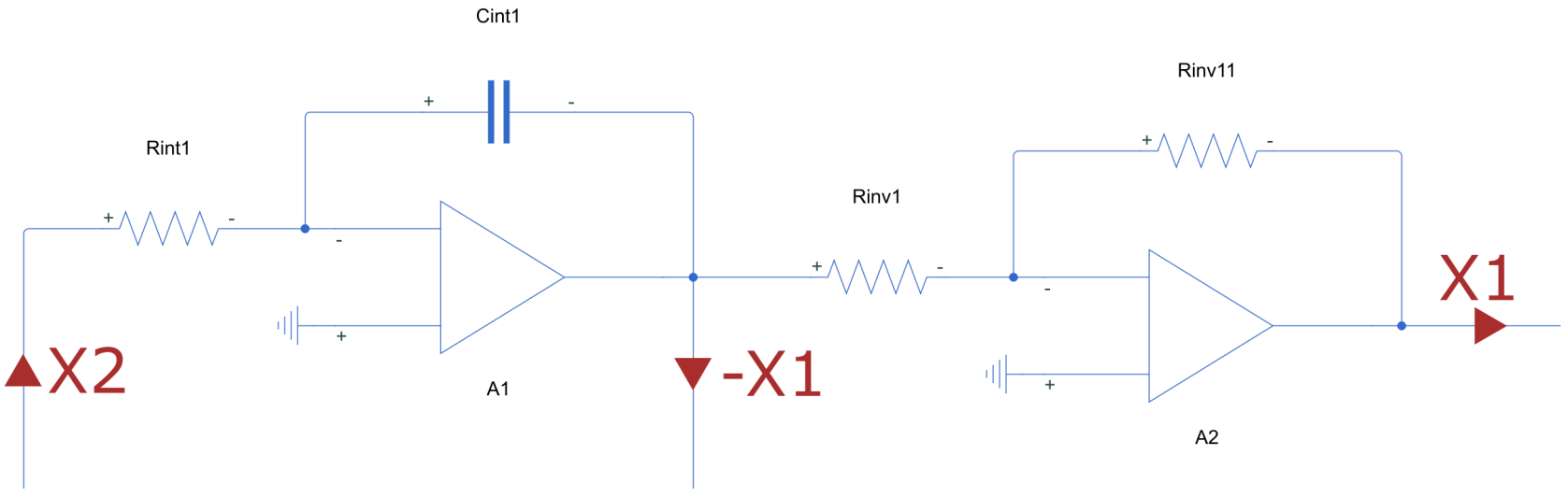
Traduzione Modello Matematico

Esempio Massa-Molla-Smorzatore - Stato x_1

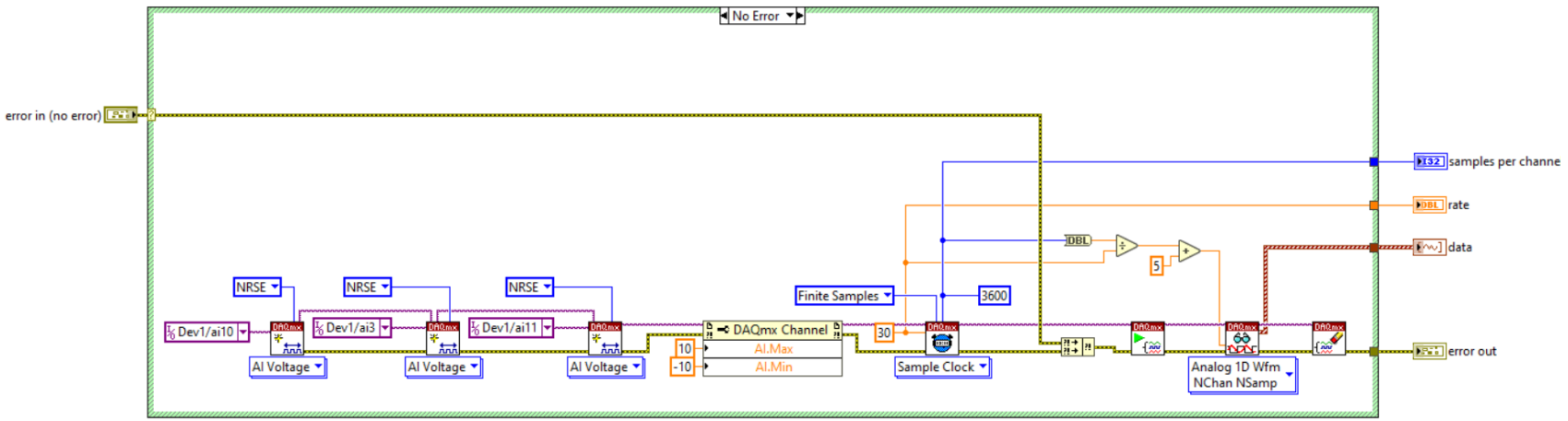
$$\dot{x_1} = x_2$$

Integratore Invertente

Amplificatore Invertente

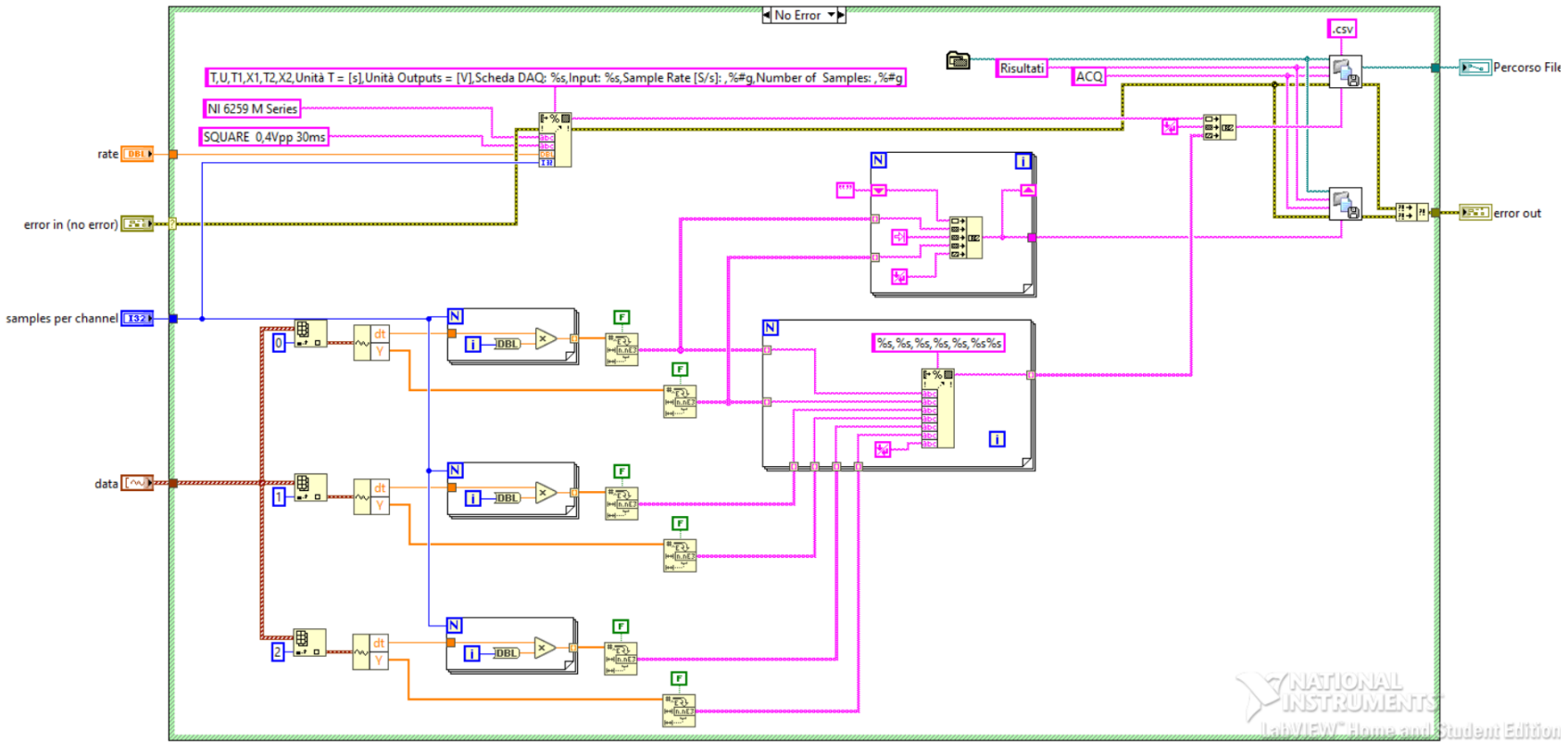


Sub VI *getSignals*

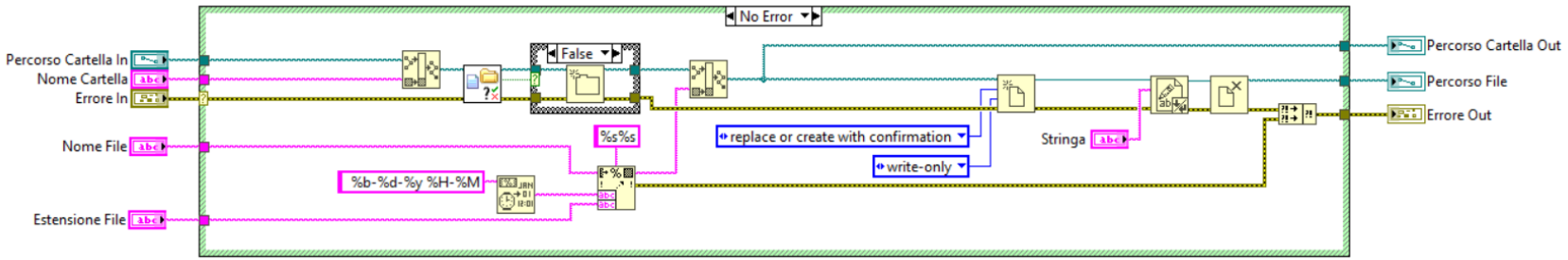


CODICE LABVIEW 2/4

Sub VI *logData*

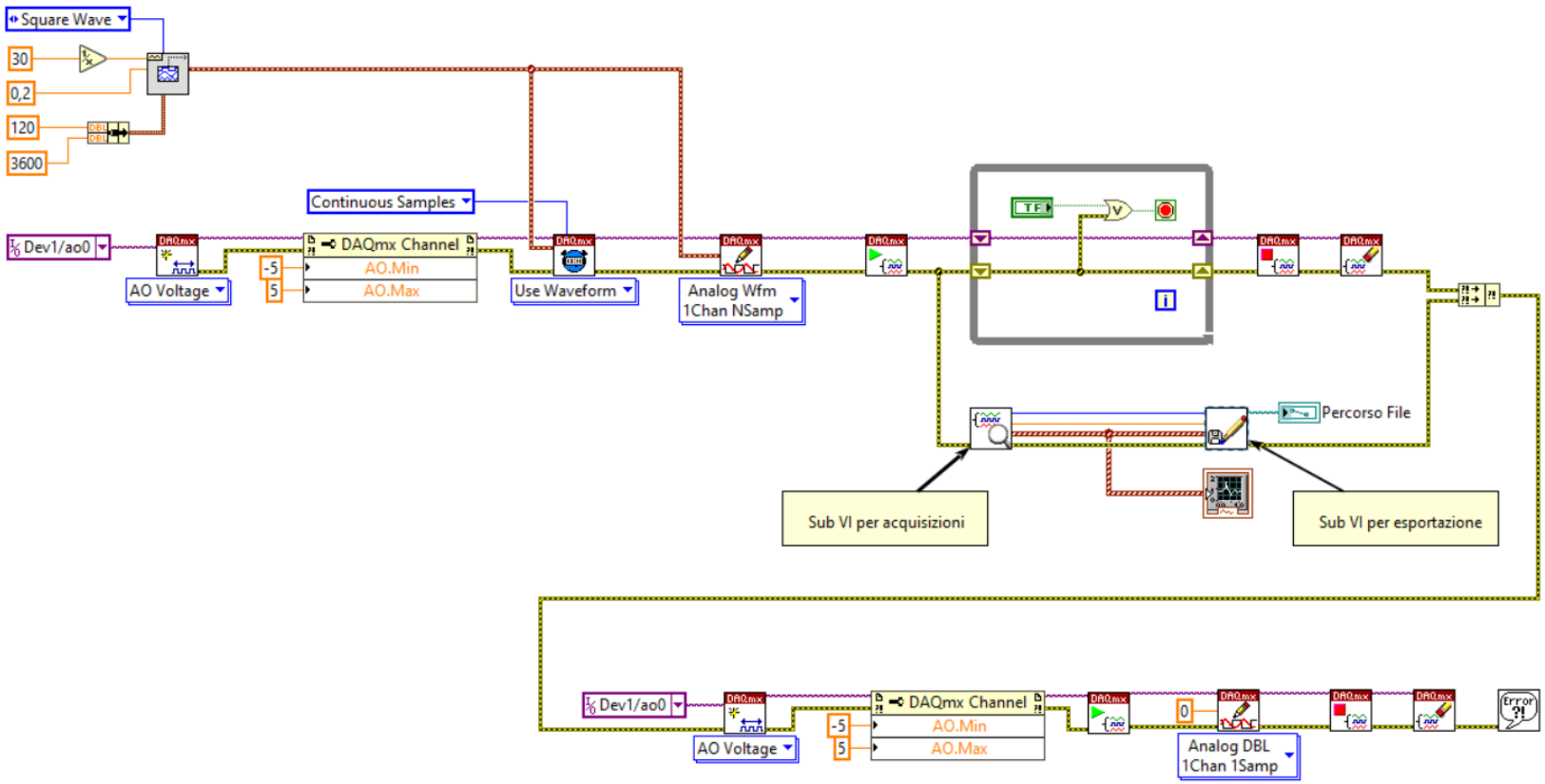


Sub VI *Save*

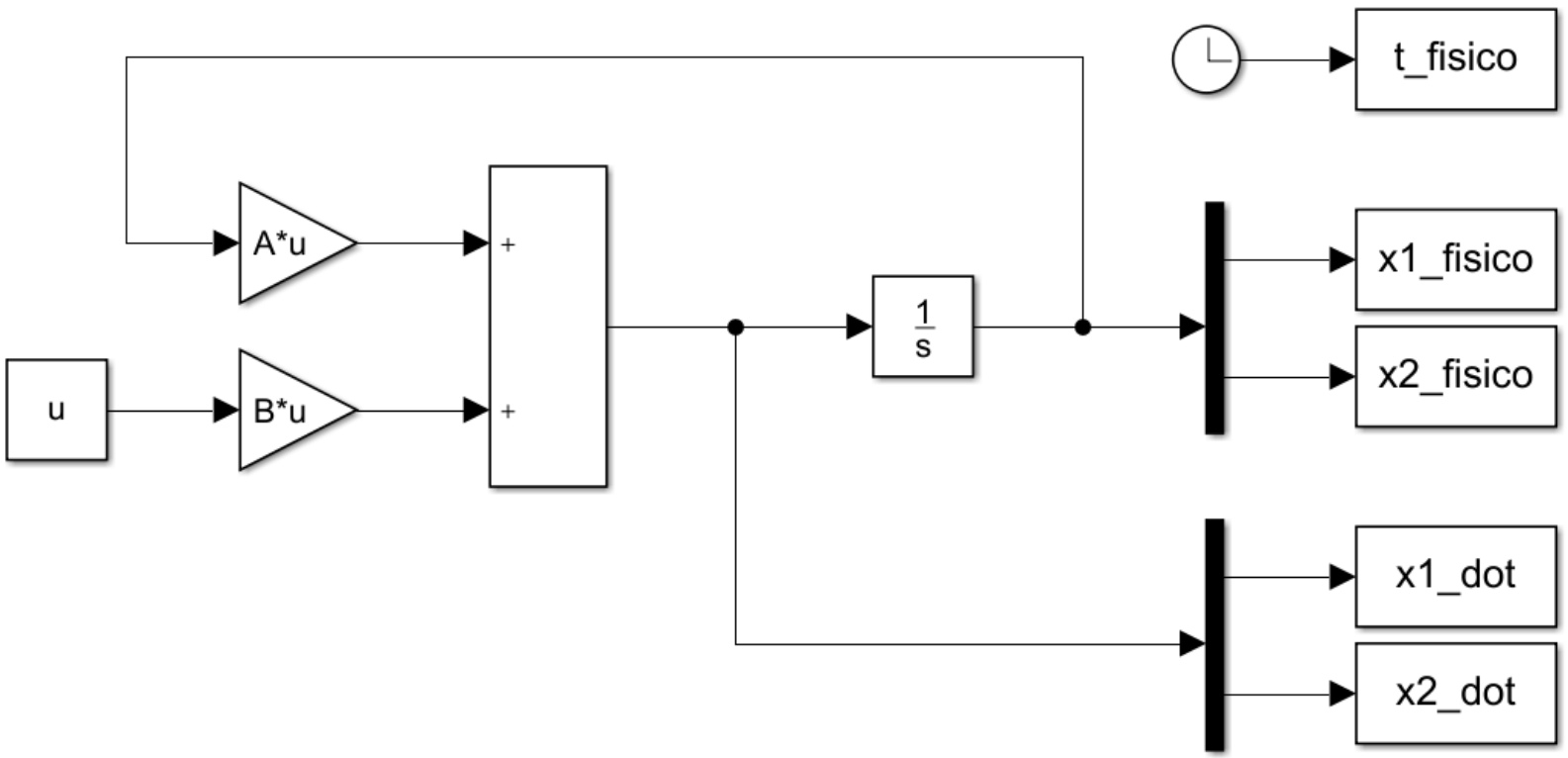

No Error
False
Percorso Cartella In
Nome Cartella
Errore In
Nome File
Estensione File
%s%s
%b-%d-%y %H-%M
replace or create with confirmation
write-only
Stringa
Percorso Cartella Out
Percorso File
Errore Out

CODICE LABVIEW 4/4

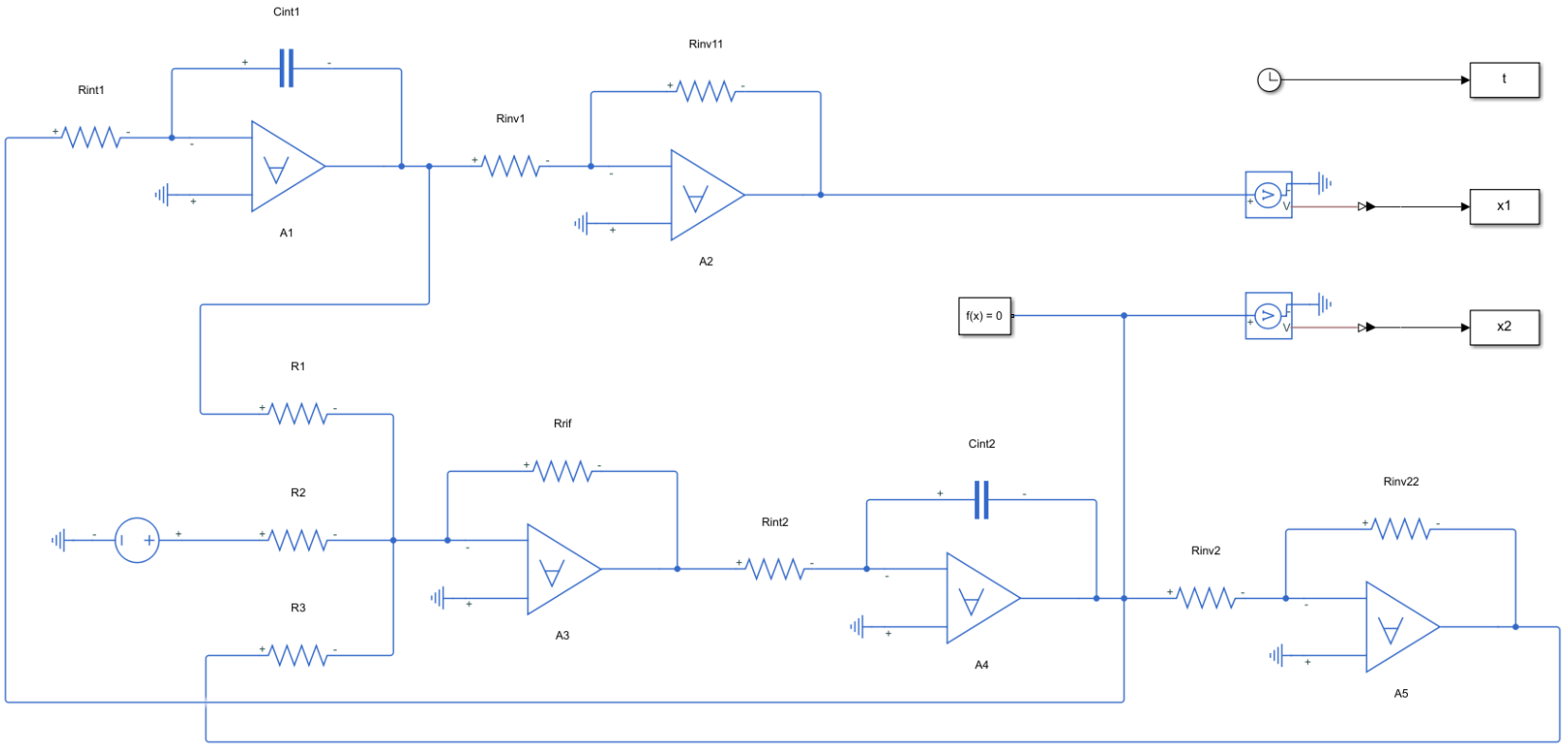
VI *Main*



Esempio di modello matematico implementato in Simulink

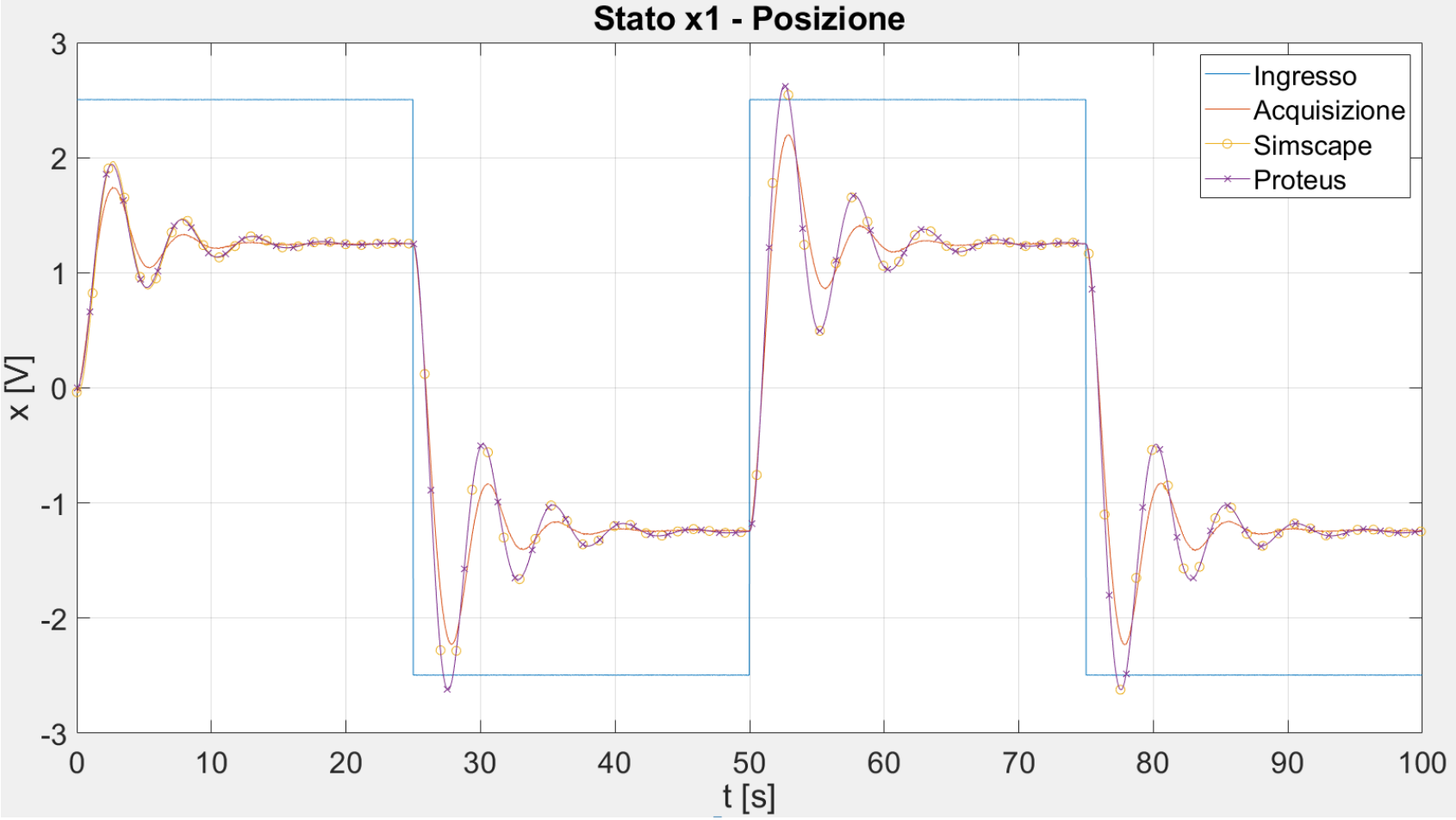


Esempio di schema Simscape per circuito elettrico



Verifica Setup Sperimentali 1/4

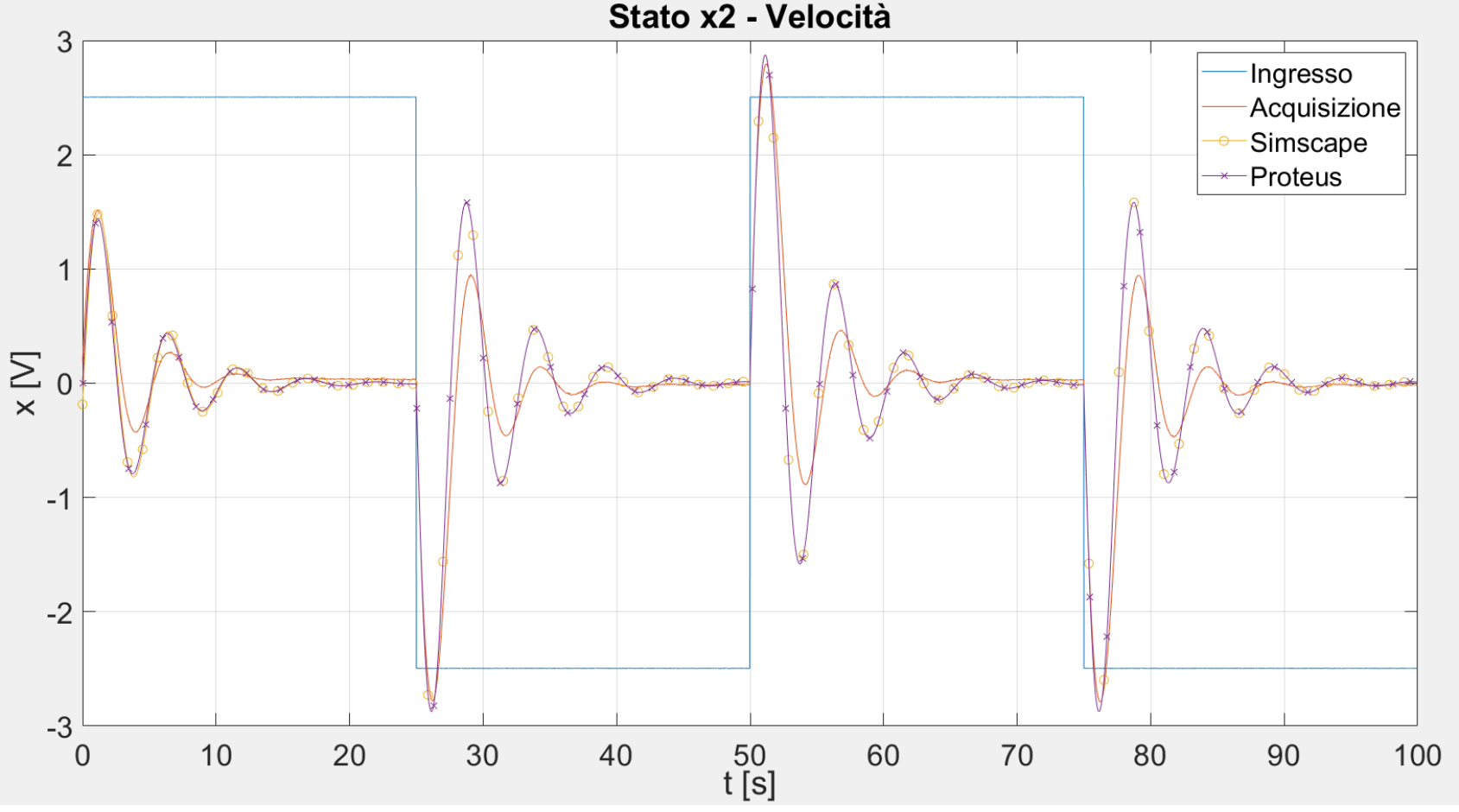
Massa-Molla-Smorzatore - Stato x_1



Ingresso/Risposta → BLU / ROSSO Simscape/Proteus → GIALLO / VIOLA

Verifica Setup Sperimentali 2/4

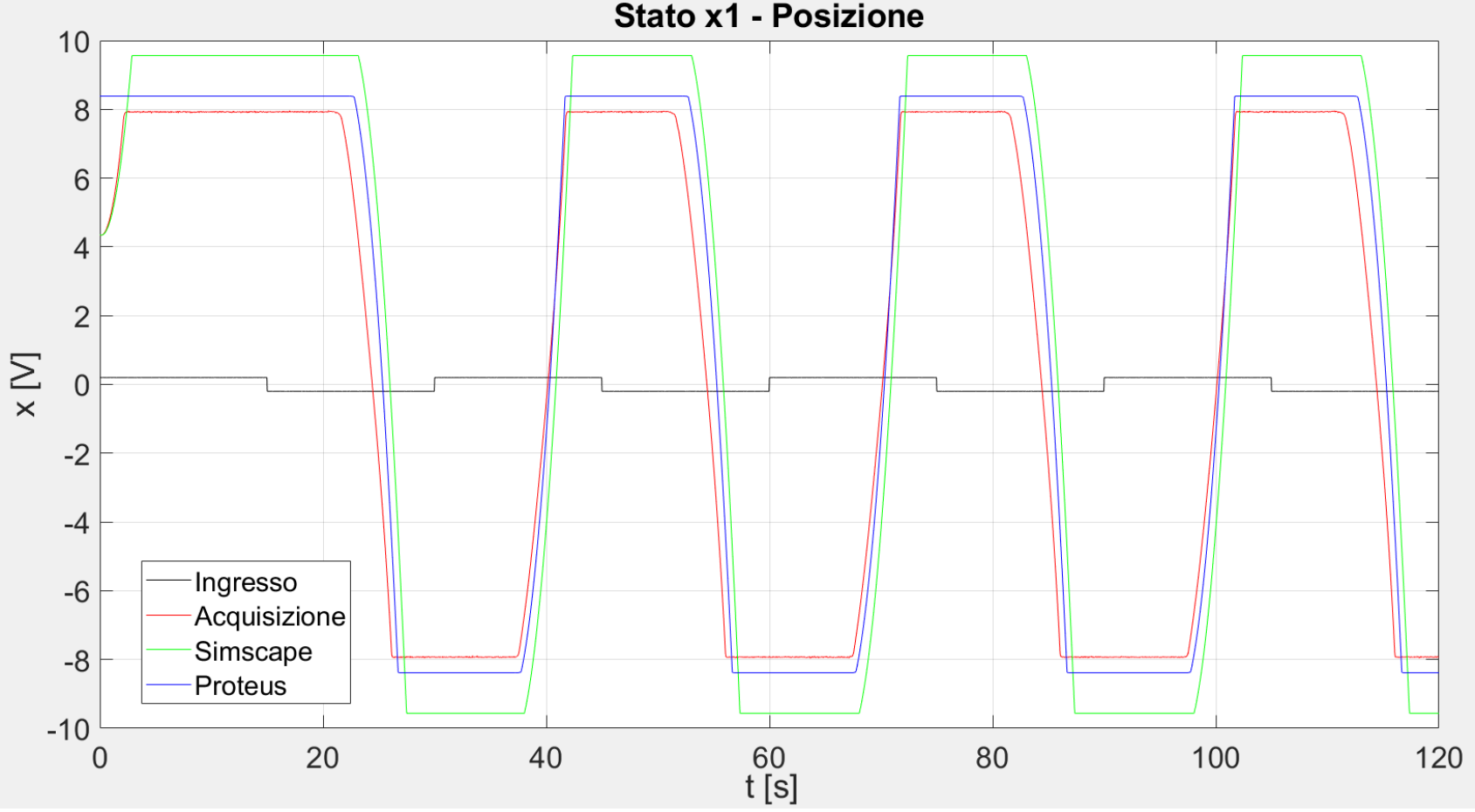
Massa-Molla-Smorzatore - Stato x_2



Ingresso/Risposta → BLU / ROSSO Simscape/Proteus → GIALLO / VIOLA

VERIFICA SETUP SPERIMENTALI 3/4

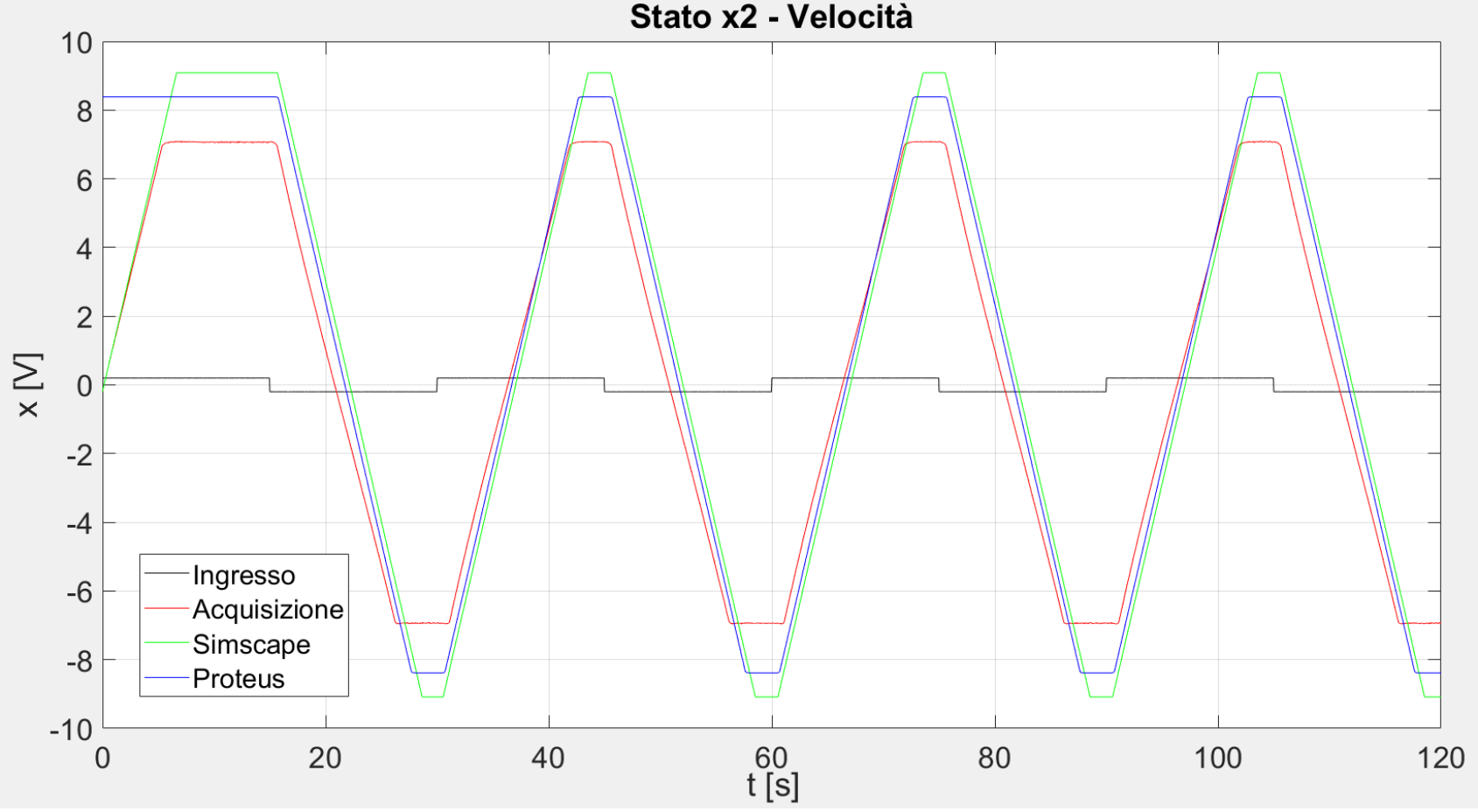
Ball-Beam - Stato x_1



Ingresso/Risposta $\rightarrow$ NERO / ROSSO Simscape/Proteus $\rightarrow$ VERDE / BLU

Verifica Setup Sperimentali 4/4

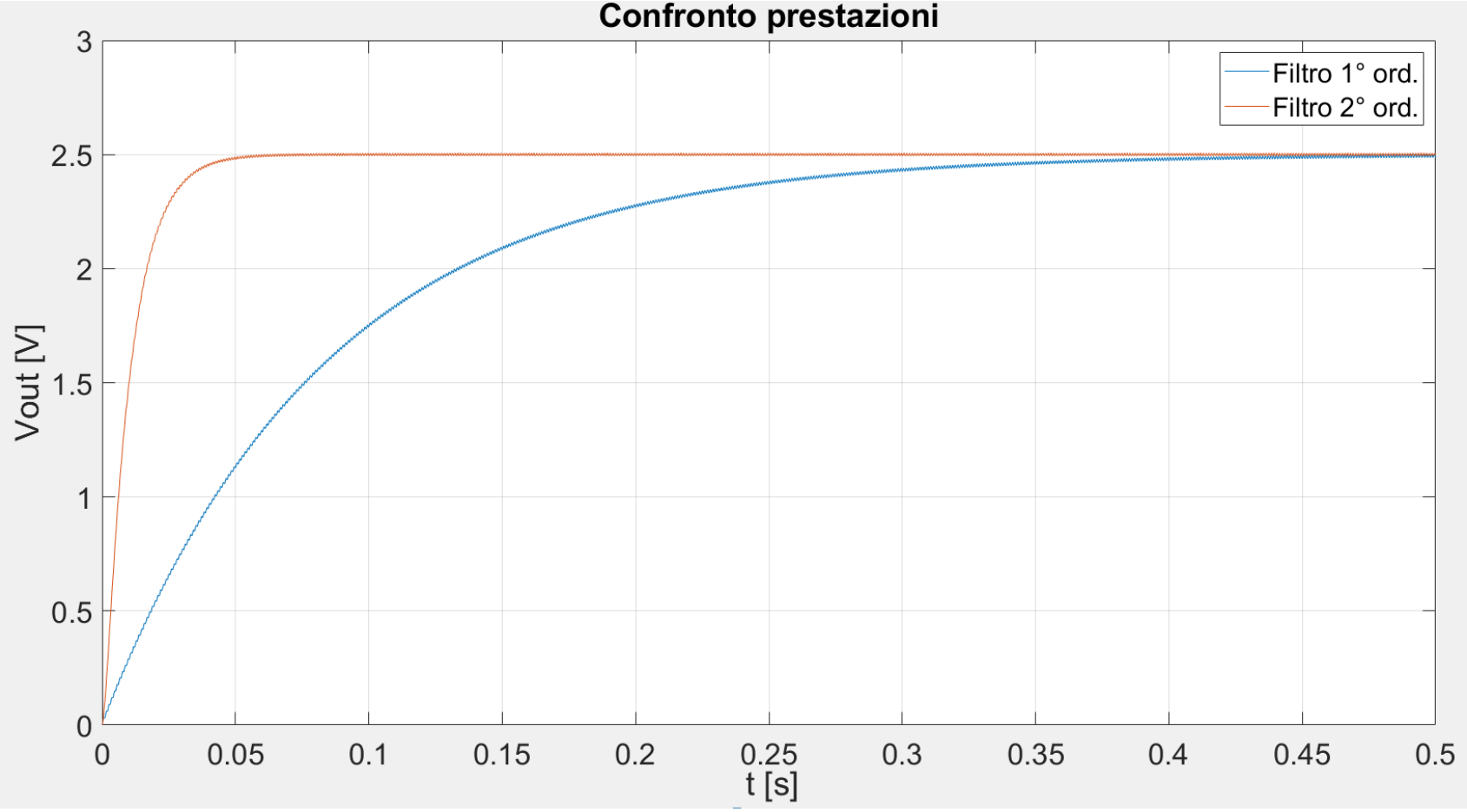
Ball-Beam - Stato x_2



Ingresso/Risposta → NERO / ROSSO Simscape/Proteus → VERDE / BLU

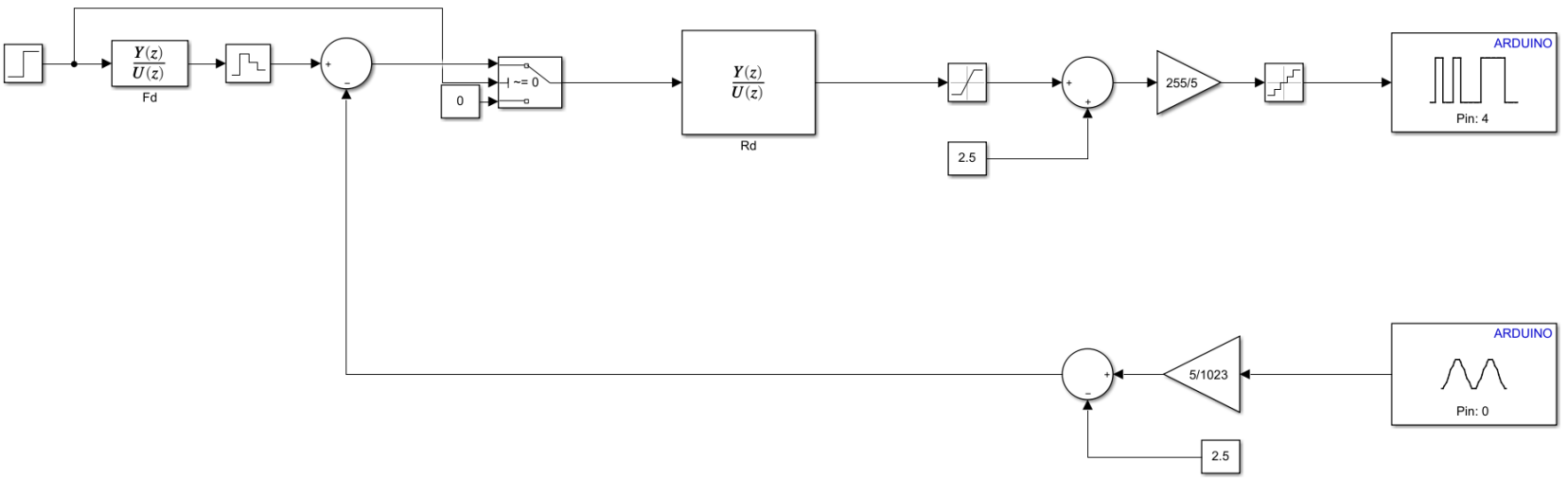
Prestazioni Filtro Passa-Basso

Risposta filtro passa-basso di primo e secondo ordine.



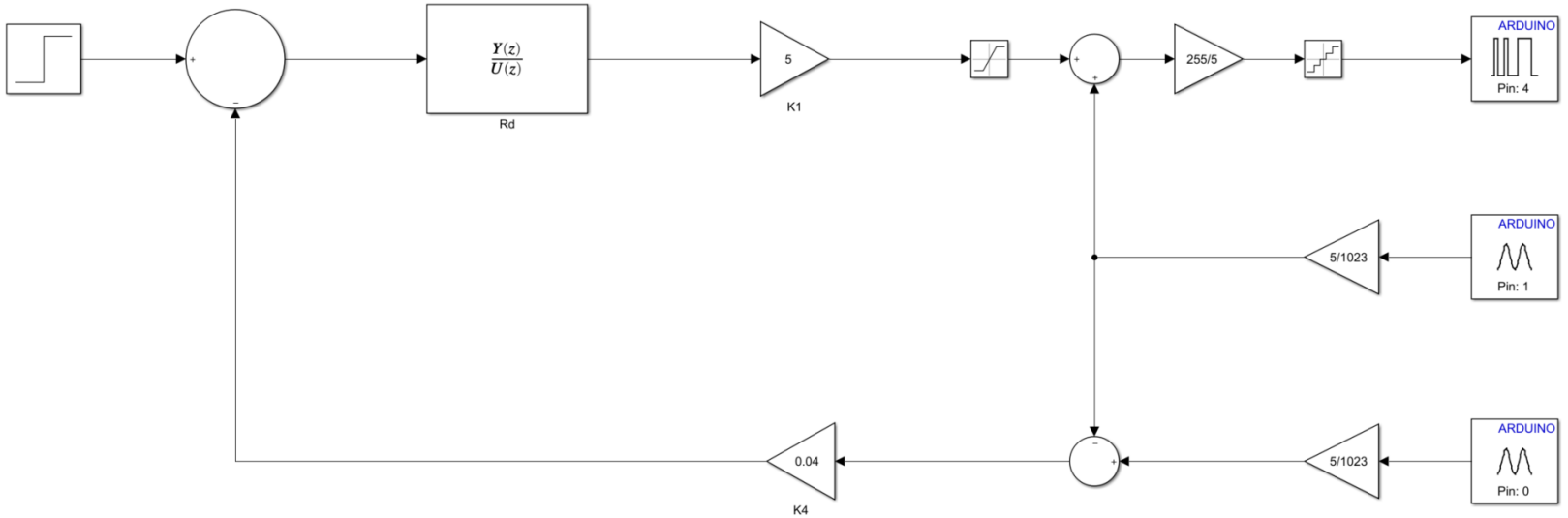
Schema di controllo Simulink 1/2

Schema di controllo Massa-Molla-Smorzatore per Arduino MEGA 2560.

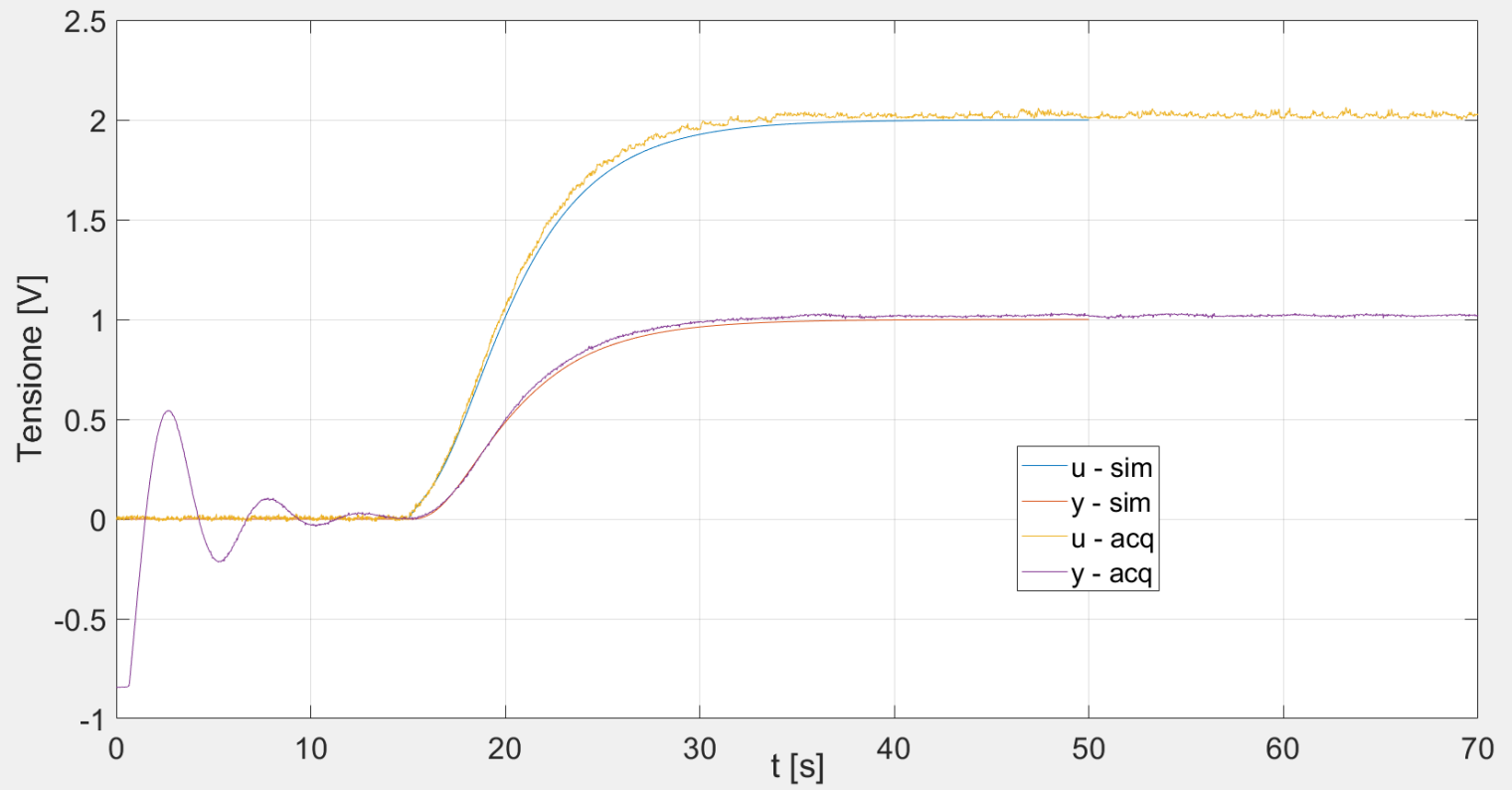


Schema di controllo Simulink 2/2

Schema di controllo Ball-Beam per Arduino MEGA 2560.

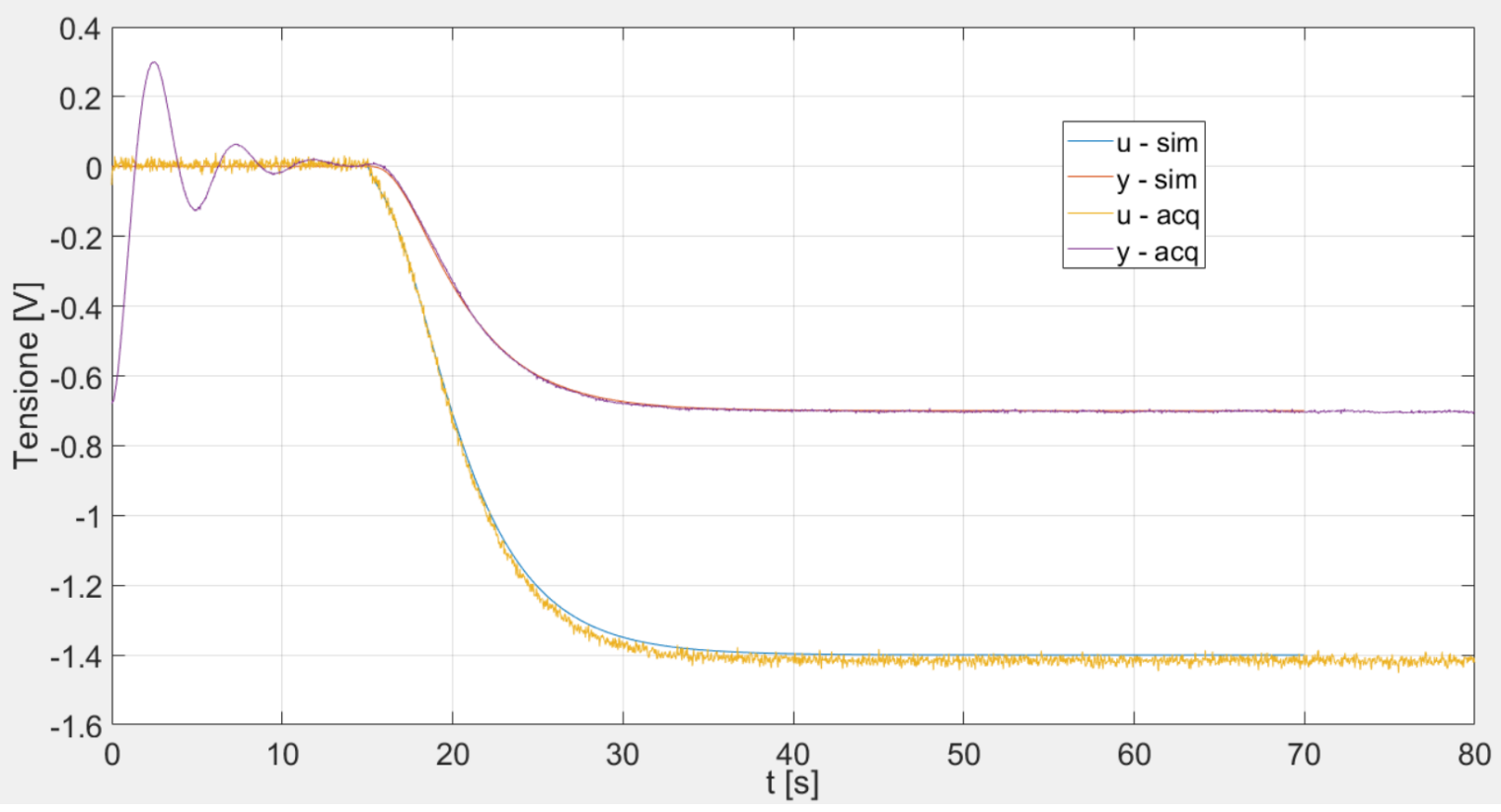


Riferimento Gradino $\Longrightarrow$ Istante applicazione = 15 s - Ampiezza gradino = 1



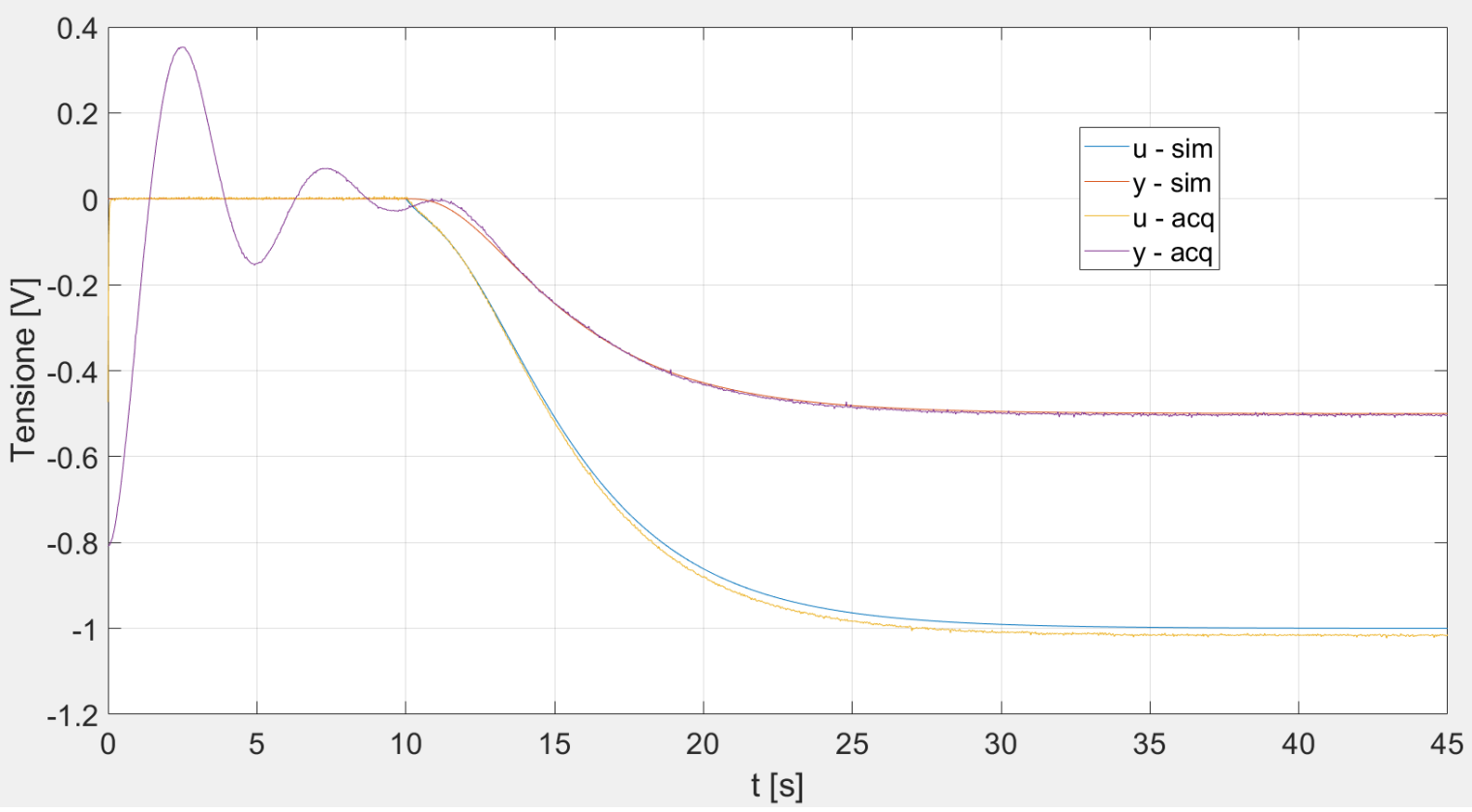
Ingresso/Uscita **Simulati** $\rightarrow$ BLU / ROSSO Ingresso/Uscita **Acquisiti** $\rightarrow$ GIALLO / VIOLA

Riferimento Gradino $\Longrightarrow$ Istante applicazione = 15 s - Ampiezza gradino = -0.7



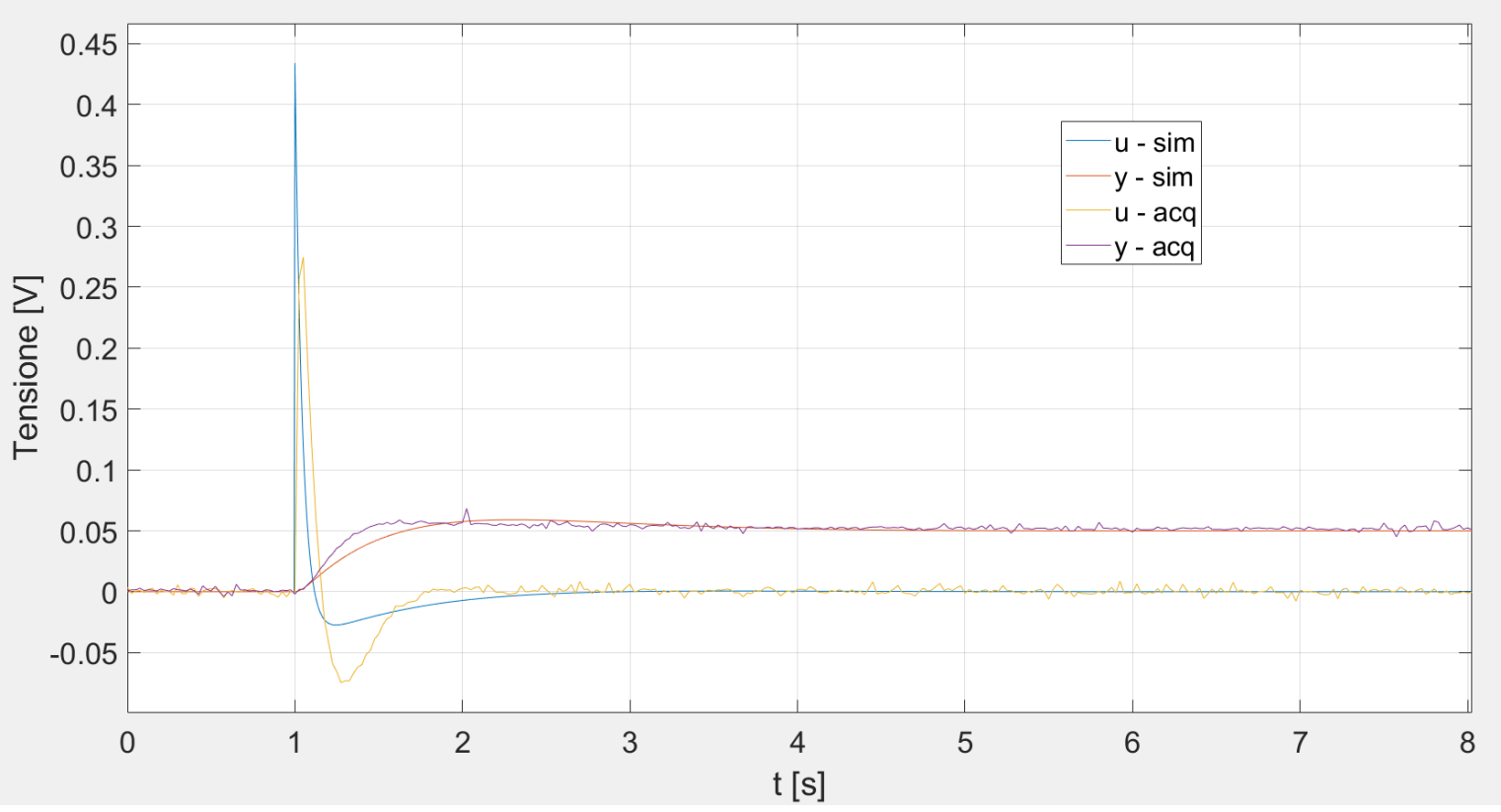
Ingresso/Uscita **Simulati** $\rightarrow$ BLU / ROSSO Ingresso/Uscita **Acquisiti** $\rightarrow$ GIALLO / VIOLA

Riferimento Gradino $\Longrightarrow$ Istante applicazione = 10 s - Ampiezza gradino = -0.5



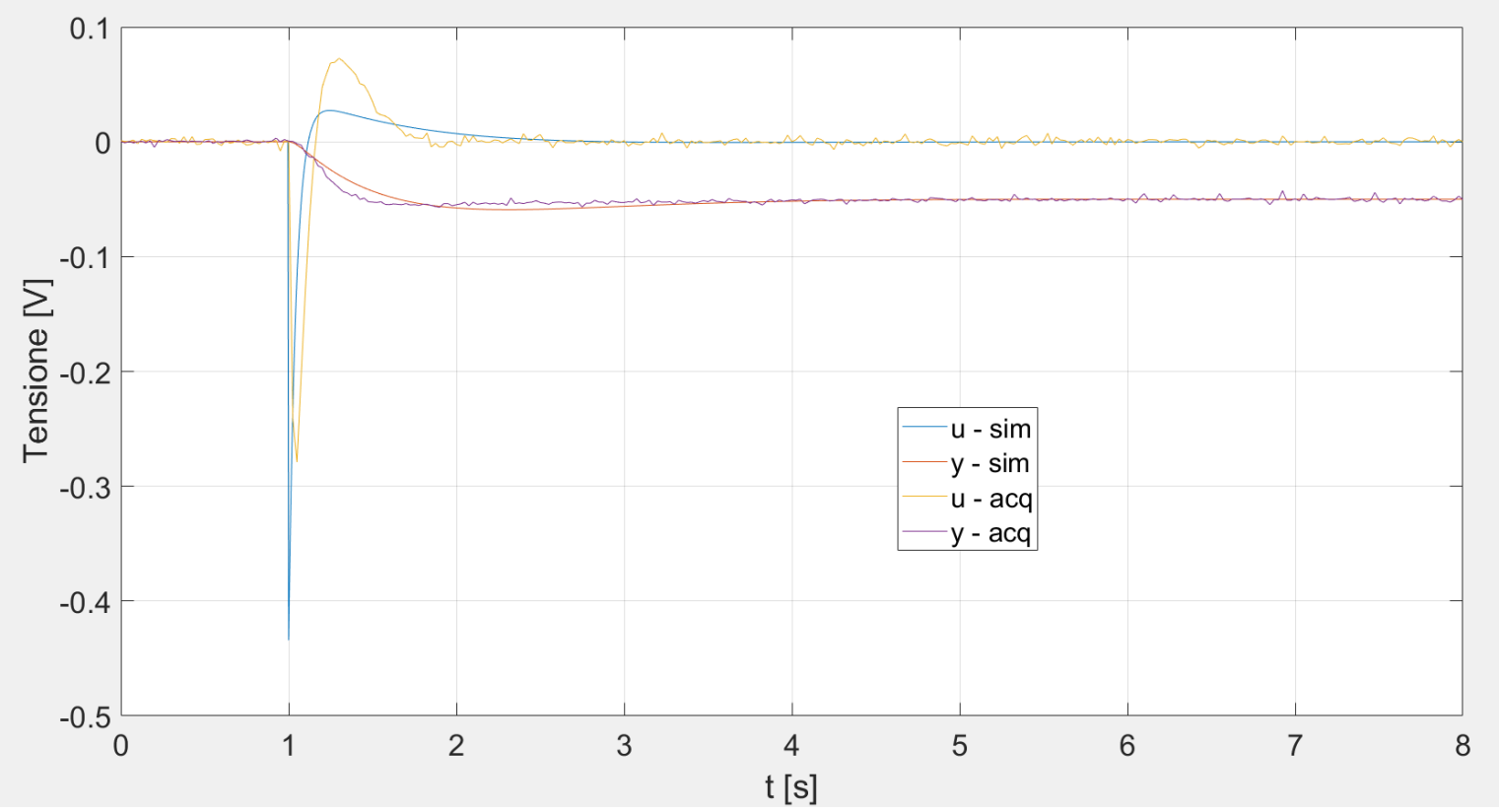
Ingresso/Uscita **Simulati** $\rightarrow$ BLU / ROSSO Ingresso/Uscita **Acquisiti** $\rightarrow$ GIALLO / VIOLA

Riferimento Gradino $\Longrightarrow$ Istante applicazione = 1 s - Ampiezza gradino = 0.05



Ingresso/Uscita **Simulati** $\rightarrow$ BLU / ROSSO Ingresso/Uscita **Acquisiti** $\rightarrow$ GIALLO / VIOLA

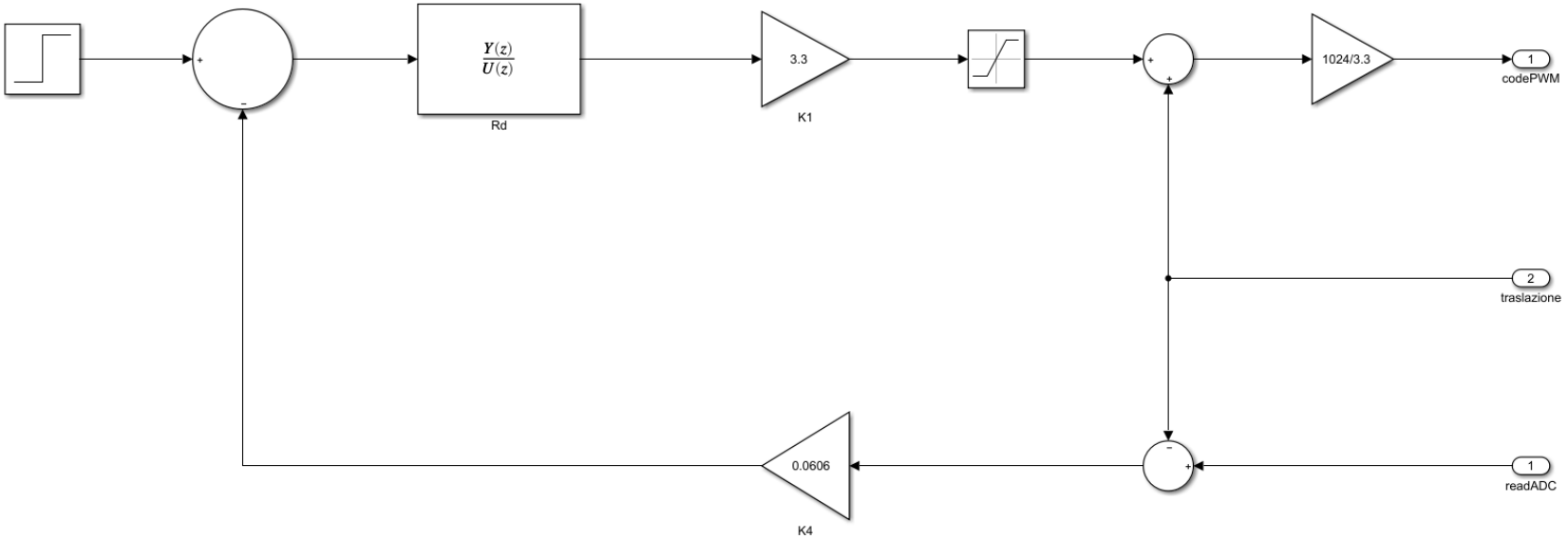
Riferimento Gradino $\Longrightarrow$ Istante applicazione = 1 s - Ampiezza gradino = -0.05



Ingresso/Uscita **Simulati** $\rightarrow$ BLU / ROSSO Ingresso/Uscita **Acquisiti** $\rightarrow$ GIALLO / VIOLA

SCHEMA DI CONTROLLO SIMULINK PER RASPBERRY

Schema di controllo Ball-Beam per Raspberry Pi 3 model B.



Compilazione ed Esecuzione 1/2

Porzione del makefile (sezione *Faster Runs Build Configuration*) modificata con l'indicazione della libreria da considerare.

```
#------------------------------------
# "Faster Runs" Build Configuration
#------------------------------------

ARFLAGS              = -r
ASFLAGS              = -c \
                       $(ASFLAGS_ADDITIONAL) \
                       $(INCLUDES)
CFLAGS               = -c \
                       -MMD -MP -MF"$(@:%.o=%.dep)" -MT"$@"  \
                       -O2
CPPFLAGS             = -c \
                       -MMD -MP -MF"$(@:%.o=%.dep)" -MT"$@"  \
                       -O2
CPP_LDFLAGS          = -lrt -lpthread -lwiringPi -ldl
CPP_SHAREDLIB_LDFLAGS  = -shared  \
                         -lrt -lpthread -lwiringPi -ldl
DOWNLOAD_FLAGS       =
EXECUTE_FLAGS        =
LDFLAGS              = -lrt -lpthread -lwiringPi -ldl
MEX_CPPFLAGS         =
MEX_CPPLDFLAGS       =
MEX_CFLAGS           =
MEX_LDFLAGS          =
MAKE_FLAGS           = -f $(MAKEFILE)
SHAREDLIB_LDFLAGS    = -shared  \
                       -lrt -lpthread -lwiringPi -ldl
```

Compilazione ed Esecuzione 2/2

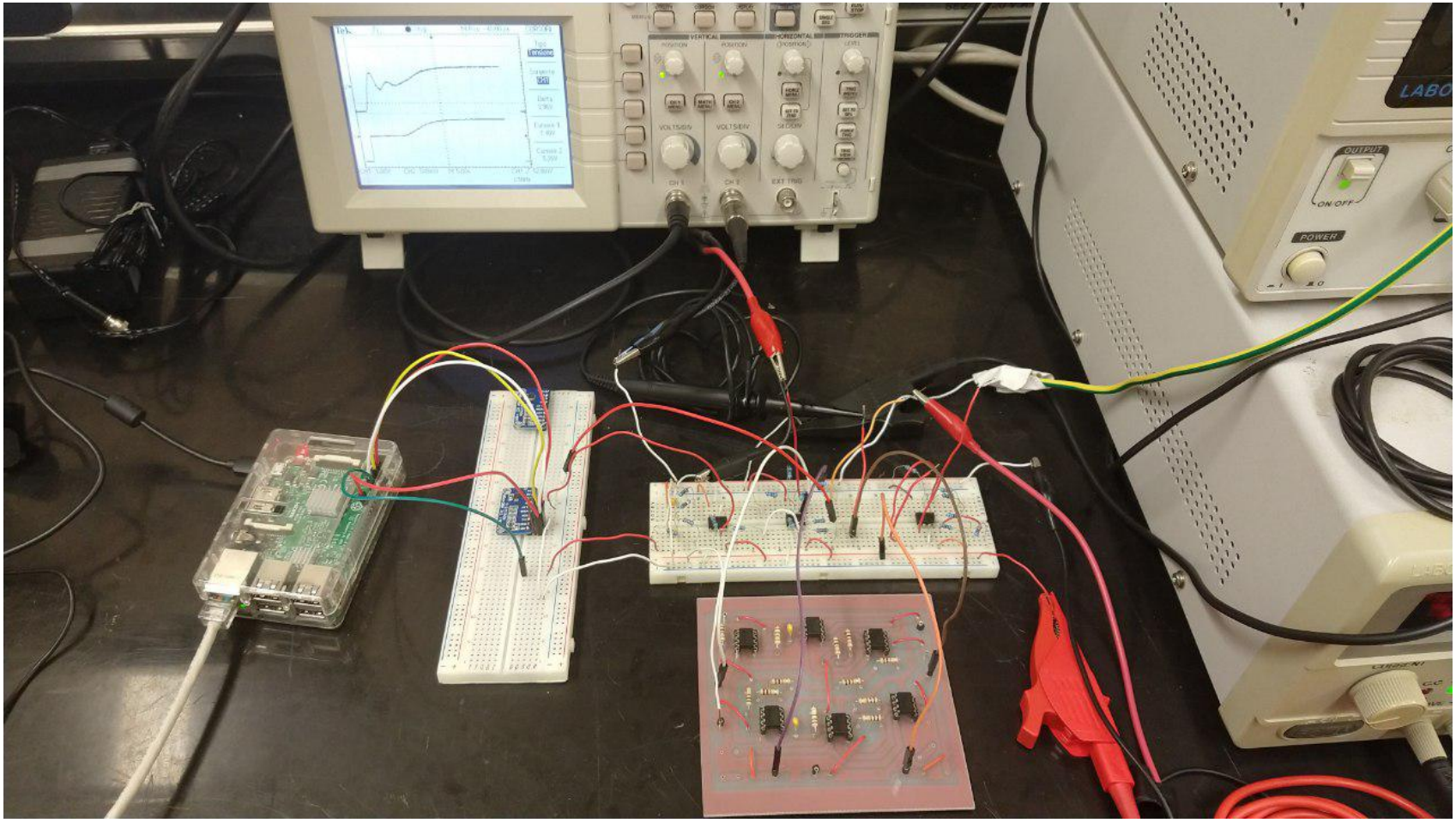
L'utility *make* è quella che si occupa della gestione della compilazione in base alle istruzioni fornite dal makefile: nella presente applicazione, si è reso necessario aggiungere al comando la dichiarazione di una variabile di ambiente utilizzata nel makefile.

```
pi@realtimepi:~/massaMollaSmorzatore/MATLAB_ws/R2019a/C/Users/Salvatore/Desktop/Tesi_Magistrale/embeddedMMS_raspControlloMMS_ert_rtw $ make -f controlloMMS.mk all MATLAB_WORKSPACE=~/massaMollaSmorzatore/MATLAB_ws/R2019a
```

L'esecuzione del comando genera, nella cartella padre di quella contenente il makefile, un eseguibile con estensione *.elf* da avviare tramite comando *sudo*. Una prima conferma della validità della procedura è ottenibile con il comando *top*.

```
pi@realtimepi: ~/ballBeam/MATLAB_ws/R2019a/C/Users/Salvatore/Desktop/Tesi_Magistrale/embeddedBB_raspPi
top - 15:00:28 up  3:23,  1 user,  load average: 1.12, 1.00, 0.77
Tasks:   1 total,   0 running,   1 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  2.4 sy,  0.0 ni, 97.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :   948908 total,   785760 free,    31868 used,   131280 buff/cache
KiB Swap:   102396 total,   102396 free,        0 used.   855064 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 1514 root      rt   0   21440   1220   1060 S   5.0  0.1   0:11.90 controlloBB.elf
```

Setup complessivo del sistema di controllo a ciclo chiuso.



Progettazione e Sviluppo Sperimentale di Sistemi di Controllo su Piattaforme Embedded

Università degli Studi **Mediterranea** di Reggio Calabria

Relatore
Prof. Valerio Scordamaglia

Candidato
Salvatore Avellino

Correlatore
Vito Antonio Nardi