UNIVERSITÀ DI PADOVA · 1222

Sviluppo di un sistema a microcontrollore per il controllo di una piattaforma di Stewart

Comunicazione digitale Simulink- STM32 e controllo attuatori

AUTORI

Bernabei Alessio - 2104152
Canola Angelo - 2101322
Gari Edoardo - 2104912
Valentini Cristian - 2109134

RELATORE

Prof. Roberto Oboe

CO-RELATRICE

Prof.ssa Giulia Michieletto

Scuola di Ingegneria · Dipartimento di Tecnica e Gestione dei Sistemi Industriali (DTG)

A.A. 2025/ 2026

Agenda

Percorso della presentazione

01

Contesto e obiettivi

La piattaforma Stewart e il problema di comunicazione

- Cos'è una piattaforma Stewart e a cosa serve
- Obiettivo: sostituire l'I/O analogico Quanser con una catena digitale
- Motivazione: hardware standard, riproducibilità, portabilità del setup

La piattaforma Stewart

Manipolatore parallelo a 6 gradi di libertà

Cos'è

- 6 attuatori lineari indipendenti
- Piattaforma superiore mobile con 6 DOF: 3 traslazioni + 3 rotazioni
- Cinematica parallela: elevata rigidezza e precisione di posizionamento
- Applicazioni: simulatori di volo, test vibrazionali, robotica, simulazione del moto ondoso

Obiettivo del progetto

Sostituire l'acquisizione tramite l'interfacciamento analogico con scheda di acquisizione Quanser con una **catena di comunicazione seriale digitale** tra:

Simulink ⇄ STM32F767ZI ⇄ Attuatori

02

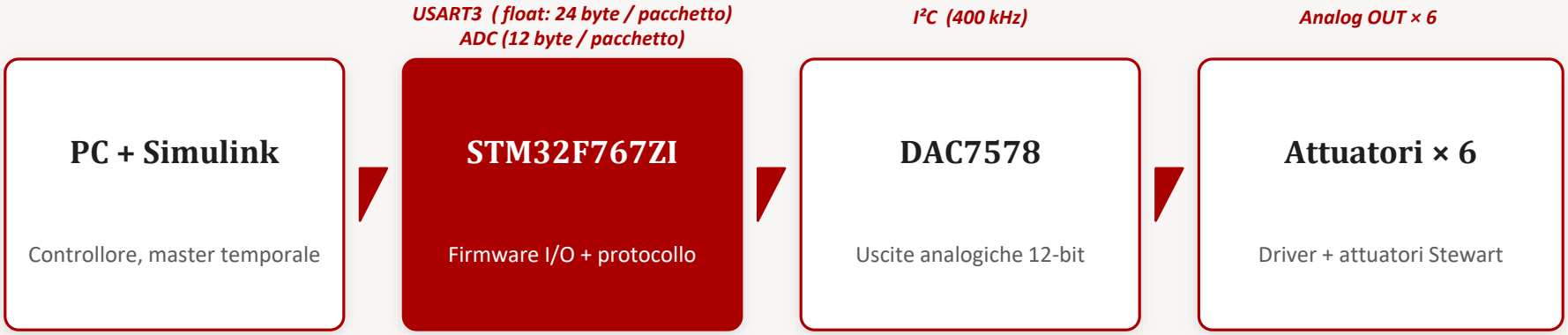
Architettura del sistema

Dallo schema Analog Input alla comunicazione digitale

- Vista d'insieme del data flow
- Innovazione
- Ciclo di comunicazione:

Architettura del sistema

Vista d'insieme del data flow



Feedback ← ADC 6-ch (DMA) letti dagli attuatori e re-inviati a Simulink via USART3

Simulink è il master del tempo: ogni ciclo di controllo è aperto da un pacchetto inviato al uC.

Vecchio vs nuovo sistema

Perché passare al digitale

SISTEMA PRECEDENTE

- Acquisizione tramite l' interfacciamento analogico con scheda di acquisizione Quanser
- Ingressi analogici diretti dagli attuatori
- Uscite analogiche dirette dalla scheda Quanser
- Hardware proprietario e costoso
- Sensibilità a rumore sui cavi analogici lunghi
- Difficoltà di scaling e portabilità del setup

SISTEMA ATTUALE

- STM32F767ZI come ponte digitale
- ADC del uC + DMA per lettura feedback (6 canali)
- Comunicazione USART3 con Simulink (Virtual COM)
- Comandi digitali tramite I^2C al DAC7578 esterno
- Setup low-cost, riproducibile, portabile
- Sincronizzazione software controllata da Simulink

Ciclo di comunicazione

Le tre fasi di ogni step di controllo

1

WAIT_CMD

Simulink → STM32

Apertura del ciclo: Simulink invia direttamente i 6 comandi (float: 32 bit, 24 byte totali).
STM32 attende con timeout.

2

APPLY_CMD

STM32 → DAC7578

Scrittura buffer + broadcast update via I²C: le 6 uscite analogiche si aggiornano simultaneamente.

3

SEND_MEAS

STM32 → Simulink

Snapshot ADC (adc_latest) inviato come 12 byte a Simulink. Il ciclo si chiude.

Il ciclo si ripete a frequenza fissa di 100 Hz imposta da Simulink.

03

Strumenti utilizzati

Hardware, ambiente di sviluppo, motivazioni delle scelte

- MATLAB / Simulink
- STM32F767ZI Nucleo
- DAC7578

MATLAB / Simulink

Ambiente di modellazione e controllo

Ruolo nel progetto

- Modellazione del controllore della piattaforma
- Generazione dei comandi CMD ad ogni step
- Elaborazione delle misure MEAS (conversione)
- Master temporale del sistema (clock del ciclo)
- Simulazione hardware-in-the-loop

Blocchi e toolbox utilizzati

Desktop Real Time

Packet Input / Output per comunicazione seriale tra Matlab e STM32

MatLab Function for Data Type Conversion

volt[6] → float[6] →uint8[24]
uint8 [12]→ volt[6]

STM32F767ZI Nucleo

Perché è stato scelto questo microcontrollore

Motivazioni della scelta

Potenza a margine

Cortex-M7 con FPU gestisce il ciclo con headroom per elaborazione futura.

Periferiche complete

ADC multi-canale con DMA, USART, I²C: tutte le periferiche necessarie native.

Nucleo-144 = ST-LINK integrato

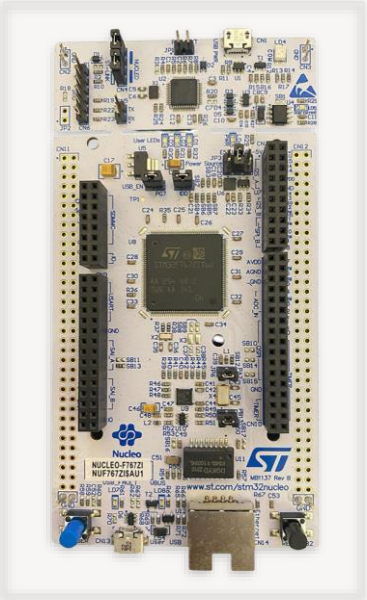
Debug + Virtual COM (USART3 su PD8/PD9) su un solo cavo micro-USB.

Ecosistema HAL / CubeMX

Configurazione periferiche assistita, portabilità del codice.

Diffusione e supporto

Documentazione, esempi, community: riduce i tempi di sviluppo.



Scheda NUCLEO-F767ZI

Specifiche chiave

CPU	ARM Cortex-M7 @ 216 MHz
FPU	Single + double precision
Cache	L1 I/D 16 KB
Flash / RAM	2 MB / 512 KB
ADC	3 × 12-bit, fino a 24 canali
DMA	2 controller, 16 stream
UART / USART	4 + 4
I²C	4 canali
Debug	ST-LINK on-board (VCOM)

DAC7578

Perché è stato scelto questo DAC

Motivazioni della scelta

8 canali su un unico chip

Di cui 6 necessari. Riduce complessità hardware.

Aggiornamento simultaneo

Modalità broadcast: tutti i canali cambiano nello stesso istante.

I²C: pochi pin

Solo SDA + SCL, indipendente dal numero di canali.

12 bit: risoluzione adeguata

Corrisponde a quella dell'ADC del micro, coerenza numerica.

Reperibilità e costo

Componente standard, ampiamente disponibile su breakout board.

Breakout DAC7578 (TSSOP-16)

Caratteristiche

Costruttore	Texas Instruments
Risoluzione	12 bit (0 – 4095)
Canali	8 uscite analogiche
Interfaccia	I²C fino a 400 kHz
Aggiornamento	Simultaneo (LDAC broadcast)
Tensione ref.	Interna / esterna
Alimentazione	2.7 – 5.5 V
Package	16-pin TSSOP
Uscite usate	6 su 8

Toolchain di sviluppo

Software utilizzati e loro ruolo

STM32CubeIDE / CubeMX

Configurazione periferiche, generazione codice HAL, compilazione.

Keil MDK

Ambiente alternativo di sviluppo firmware, usato per debug avanzato.

MATLAB + Simulink (Real-Time)

Modellazione del controllore, esecuzione in real-time del ciclo.

Instrument Control Toolbox

Blocchi Packet Input/Output per la comunicazione seriale.

Logic Analyzer + Oscilloscopio

Debug hardware di I²C, USART, ADC e uscite DAC.

Datasheet + Reference Manual

STM32F767ZI, DAC7578 datasheet, application note ST.

04

Periferiche del uC

Configurazione e ruolo di ADC, DMA, USART, I²C

- ADC + DMA in scan mode
- USART3 come Virtual COM ST-LINK
- I²C verso DAC7578

Mappa delle periferiche

Cosa fa ogni blocco del uC nel progetto

ADC + DMA

6 canali scan mode

USART3

Virtual COM ↔ Simulink

STM32F767ZI

Cortex-M7 @ 216 MHz

Cache L1 · FPU · DMA

GPIO / Clock

LED user, HSE 25 MHz

I²C

SCL PB8 / SDA PB9 → DAC7578

ADC + DMA

Lettura continua dei 6 feedback in scan mode

Configurazione ADC

- Risoluzione: 12 bit (0 – 4095)
- Modalità scan multi-canale: 6 canali in sequenza
- Modalità continua: conversioni avviate automaticamente
- DMA in modalità circolare: buffer riempito senza intervento CPU
- Callback ADC a fine sequenza: DMA buf → adc_latest

Perché DMA + scan mode?

- La CPU non deve leggere manualmente ogni conversione
- Le 6 misure arrivano in un unico blocco coerente
- Il main loop trova sempre un adc_latest aggiornato
- Nessun jitter dovuto ad interrupt di lettura

Real-Time temporale

Le 6 conversioni impiegano ~15 µs a 21 MHz ADC clock: sempre pronte prima del prossimo ciclo Simulink (10 ms).

Mapping degli ingressi ADC

Ordine dei canali nel buffer

Indice buffer	Pin fisico	Canale ADC	Segnale sorgente
`adc_latest[0]`	PA3	`ADC_CHANNEL_3`	Feedback attuatore 1
`adc_latest[1]`	PC0	`ADC_CHANNEL_10`	Feedback attuatore 2
`adc_latest[2]`	PC3	`ADC_CHANNEL_13`	Feedback attuatore 3
`adc_latest[3]`	PA5	`ADC_CHANNEL_5`	Feedback attuatore 4
`adc_latest[4]`	PA6	`ADC_CHANNEL_6`	Feedback attuatore 5
`adc_latest[5]`	PC2	`ADC_CHANNEL_12`	Feedback attuatore 6

L'ordine nel buffer segue l'ordine dei rank di conversione configurati in CubeMX.

USART3 – Virtual COM ST-LINK

Canale seriale con il PC

Configurazione

Pin TX / RX	`PD8 / PD9`
Alternate Function	`AF7`
Baud rate	`115200 bps`
Formato	`8 – N – 1`
Controllo di flusso	`Nessuno`
Modalità	`Bloccante con timeout`
Percorso fisico	`ST-LINK → USB → PC`

Vantaggi Virtual COM

- Un solo cavo USB per debug + comunicazione
- Nessun modulo esterno FT232
- Driver ST-LINK per il debug
- Compare come porta COM standard in Simulink

Robustezza software

Ricezione con timeout: se il pacchetto non arriva entro N ms, la UART viene ripulita e si torna in attesa.

I²C – Bus verso DAC7578

Comunicazione digitale coi convertitori

Configurazione

SCL	`PB8 (AF4, open-drain)`
SDA	`PB9 (AF4, open-drain)`
Velocità	`400 kHz (Fast Mode)`
Address DAC7578	`0x4c`
Pull-up	`Integrate su breakout`
Modo API HAL	`HAL_I2C_Master_Transmit`
DMA I²C	`Non usato (blockig)`

Cablaggio fisico

```
PB8  (STM32)  → SCL  (DAC7578)
PB9  (STM32)  → SDA  (DAC7578)
GND  (STM32)  → GND  (DAC7578)
3V3  (STM32)  → VCC  (DAC7578)
LDAC del DAC collegato a GND
```

- *SDA con SDA, SCL con SCL*
- *LDAC collegato a GND per trasferire i valori dei registri alle uscite analogiche del DAC*

05

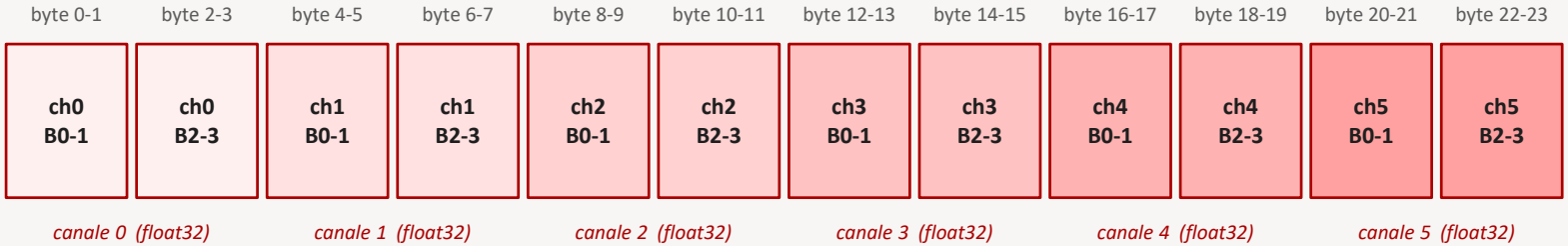
Protocollo di comunicazione

Formato pacchetti e logica del software

- Float32 (6 × 24 Byte)
- Uint16 (6 x 16 Byte)
- Fasi WAIT_CMD, APPLY_CMD, SEND_MEAS
- Simulink master del tempo

Formato del pacchetto cmd

24 byte = 6 canali × float32 IEEE-754 — dominio [−5.0, +5.0] V



Little-endian

IEEE-754 float32, byte LSB per primo.

Ordine fisso dei canali

Sempre indice 0 → 5, mai skip.

Nessun header

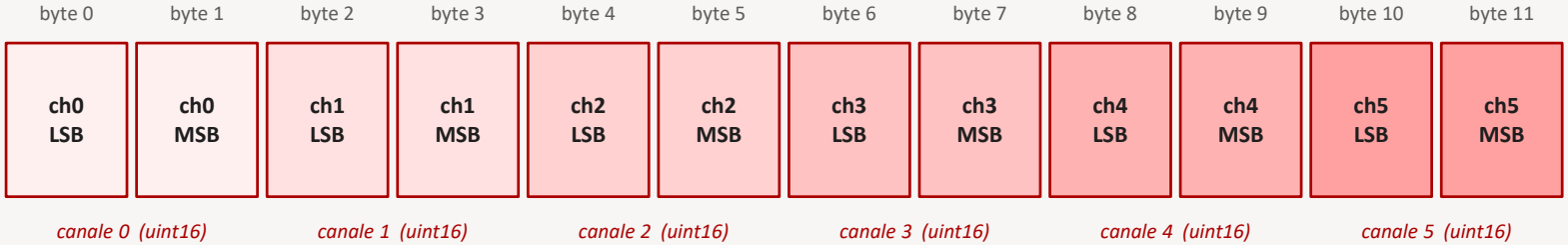
24 byte netti (6 × 4). Sincronia gestita a livello superiore.

Formato diverso Tx/Rx

CMD (Rx da Simulink) e MEAS (Tx verso Simulink) hanno layout diverso.

Formato del pacchetto MEAS

12 byte = 6 canali × uint16 — valori ADC grezzi [0, 4095]



Little-endian (LSB first)

Ogni uint16: byte basso per primo, poi alto.

Ordine fisso dei canali

Sempre indice 0 → 5, mai skip.

Nessun header

12 byte netti (6 × 2). Valori ADC grezzi, senza scaling.

Scaling lato Simulink

STM32 invia i grezzi; la conversione in volt avviene in Simulink.

Fase 1 — WAIT_CMD

Simulink apre il ciclo con i comandi

1

Chi manda cosa

- Simulink invia 24 byte = 6 × float32 (IEEE-754, dominio [−5.0, +5.0] V).
- STM32 attende in HAL_UART_Receive con UART_CMD_TIMEOUT_MS; al successo → cmd_f[0..5].
- Scalatura per canale i: u'[i] = (2047 / 5.0) · cmd_f[i] → intero in [−2047, +2047].
- Codice DAC 12-bit (uint16): u_DAC[i] = 2048 + u'[i] → [1, 4095] (2048 = metà scala, neutro).
- Catena analogica: DAC7578 [0, 3.3 V] → scheda di condizionamento → pistoni.

In caso di timeout

UART_Reset() ripulisce il buffer di ricezione rx e poi si riprova.

Perché Simulink apre col CMD

Simulink è il master del tempo: avvia ogni ciclo di comunicazione e definisce il ritmo di aggiornamento dei comandi della piattaforma.

Fase 2 — APPLY_CMD

Scrittura dei comandi al DAC7578

2

Sequenza I²C in due passi

- Per ogni canale 0..5: DAC7578_WriteBuffer(i, cmd_buf[i]) → scrive nel buffer interno.
- Alla fine: DAC7578_UpdateAll() → comando broadcast, tutte le uscite si aggiornano insieme.
- Ogni HAL_I2C_Master_Transmit è controllato: al primo errore → safe mode.
- Le uscite analogiche cambiano solo al UpdateAll finale.

Gestione errori I²C

- NACK dal DAC o bus stuck → UART_Reset().
- Il ciclo viene abortito con “continue”: si torna in WAIT_CMD.
- Metà scala (2048) = comando neutro: nessun movimento brusco.

Fase 3 — SEND_MEAS

STM32 chiude il ciclo con le misure ADC

3

Cosa succede sullo STM32

- Snapshot in sezione critica: __disable_irq() → copia adc_latest → meas_buf → __enable_irq().
- Serializzazione dei 6 uint16 in 12 byte (Uint16ArrayToBytes).
- Trasmissione con HAL_UART_Transmit su USART3.
- Simulink riceve tramite Packet Input e usa le misure nel prossimo ciclo.

Scaling lato Simulink

- Valori grezzi ADC: 0 – 4095 (12 bit).
- Conversione a tensione: guadagno = 3.3 / 4095 → valori in volt.
- Dopo lo scaling entrano nei blocchi Simulink LPF e volt2disp (da tensione a spostamento).

06

Codice

Firmware STM32 e modello Simulink

- Struttura del main loop
- Buffer e callback
- Driver DAC7578
- Modello Simulink lato PC

Struttura del main loop

Pseudo-codice del ciclo principale

```
while (1) {
  /* 1) WAIT_CMD ------------------------------------------------------------*/
  s = HAL_UART_Receive(&huart3, uart_rx_buf,MATLAB_PACKET_BYTES,UART_CMD_TIMEOUT_MS);
  if (s != HAL_OK) {
    UART_Reset();
    continue;
  }

  /* 2) APPLY_CMD -----------------------------------------------------------*/
  BytesToUint16Array(uart_rx_buf, DAC_cmd, N_CYLINDERS);
  if (Write_DAC(DAC_cmd, N_CYLINDERS) != HAL_OK)
    continue;

  /* 3) SEND_MEAS -----------------------------------------------------------*/
  __disable_irq();
  for (i = 0; i < N_CYLINDERS; i++)
    meas_buf[i] = adc_latest[i];
  __enable_irq();

  Uint16ArrayToBytes(meas_buf,uart_tx_buf, N_CYLINDERS);

  s=HAL_UART_Transmit(&huart3,uart_tx_buf, STM_MEAS_PACKET_BYTES,UART_TX_TIMEOUT_MS);
  if (s != HAL_OK) {
    UART_Reset();
    continue;
  }
}
```

1 WAIT_CMD

Riceve i 6 comandi (24 byte float) da Simulink. Su timeout: UART_Reset() e nuovo ciclo.

2 APPLY_CMD

Write_DAC() incapsula 6 write-buffer + 1 UpdateAll con fallback DAC safe.

3 SEND_MEAS

Snapshot ADC atomico (__disable_irq) e invio 12 byte via USART3 a Simulink.

Buffer principali

Dove vivono i dati nel firmware

Buffer	Tipo	Prodotto da	Usufruito da
`adc_dma_buf[6]`	`uint16_t volatile`	DMA (continuo)	ADC callback
`adc_latest[6]`	`uint16_t volatile`	ADC callback (snapshot)	Main loop
`uart_rx_buf[24]`	`uint8_t`	HAL_UART_Receive	BytesToUint16Array
`uart_tx_buf[12]`	`uint8_t`	Uint16ArrayToBytes	HAL_UART_Transmit
`meas_buf[6]`	`uint16_t`	Copia da adc_latest	Uint16ArrayToBytes
`DAC_cmd[6]`	`uint16_t`	Deserial. + saturazione	DAC7578 (I^2C)

`adc_latest` è snapshot coerente, **`DAC_cmd`** e **`meas_buf`** ponte fra bytes e uint16.

Callback ADC

Da DMA a snapshot coerente

```
/* Callback ADC: fine sequenza scan  */
/* dei 6 canali. Fotografa il buffer */
void HAL_ADC_ConvCpltCallback(
        ADC_HandleTypeDef *hadc)
{
  if (hadc->Instance == ADC1) {
    for (uint8_t i = 0; i < N_CYLINDERS; i++)
      adc_latest[i] = adc_dma_buf[i];
  }
}
```

Punti chiave

Regione MPU non-cacheable

if(hadc == ADC1): la stessa callback può servire più ADC.

Copia esplicita

for-loop su N_CYLINDERS: nessun memcpy, snapshot dei 6 uint16.

Regione MPU non-cacheable

Buffer DMA in regione MPU non-cacheable: nessuna InvalidateDCache.

Callback breve

Zero I/O e zero cache-op: rientro immediato nel main loop.

adc_latest e adc_dma_buf sono volatile: il compilatore non può ottimizzare le letture.

Driver DAC7578

Scrittura di un canale e update simultaneo

```
static HAL_StatusTypeDef Write_DAC(const uint16_t *value, uint8_t n)
{
    HAL_StatusTypeDef status = HAL_OK;

    for (uint8_t i = 0; i < n; i++)
    {
        status = DAC_WriteBuffer(i, value[i]);
        if (status != HAL_OK)
        {
            DAC_SetAllSafe();
            UART_Reset();
            return status;
        }
    }

    status = DAC_UpdateAll();
    if (status != HAL_OK)
    {
        DAC_SetAllSafe();
        UART_Reset();
    }
    return status;
}
```

Come funziona

Indirizzo DAC7578

0x4C (7-bit): dipende dai pin A0/A1 sul breakout board.

Formato dati 12 bit

MSB in d[1], LSB nel nibble alto di d[2] (allineato a sinistra).

Saturazione integrata

SaturateDACValue chiamato dentro DAC_WriteBuffer, non nel main loop.

Timeout finito

I2C_TX_TIMEOUT_MS al posto di HAL_MAX_DELAY: il main loop non si blocca.

Modalità sicura del DAC

Cosa fare quando qualcosa va storto

Quando si attiva

Timeout UART su CMD

Simulink si blocca, non arriva il pacchetto CMD.

Errore I²C

NACK dal DAC, bus stuck, disconnessione.

Startup del firmware

Prima di ricevere qualsiasi comando valido.

Watchdog di sicurezza

Rilevata anomalia nel ciclo di controllo.

DAC_SetAllSafe()

```
static void DAC_SetAllSafe(void)
{
  HAL_StatusTypeDef s;
  for (uint8_t i = 0; i < N_CYLINDERS; i++) {
    s = DAC_WriteBuffer(i, DAC_MID_VALUE);
    if (s != HAL_OK) return;
  }
  (void)DAC_UpdateAll();
}
```

- Porta tutti i 6 canali a metà scala (2048).
- Convenzione: metà scala = comando nullo/neutro.
- Evita movimenti bruschi in caso di fault.
- Chiamata anche allo startup del micro.

Modello Simulink

Catena Rx / Tx lato PC

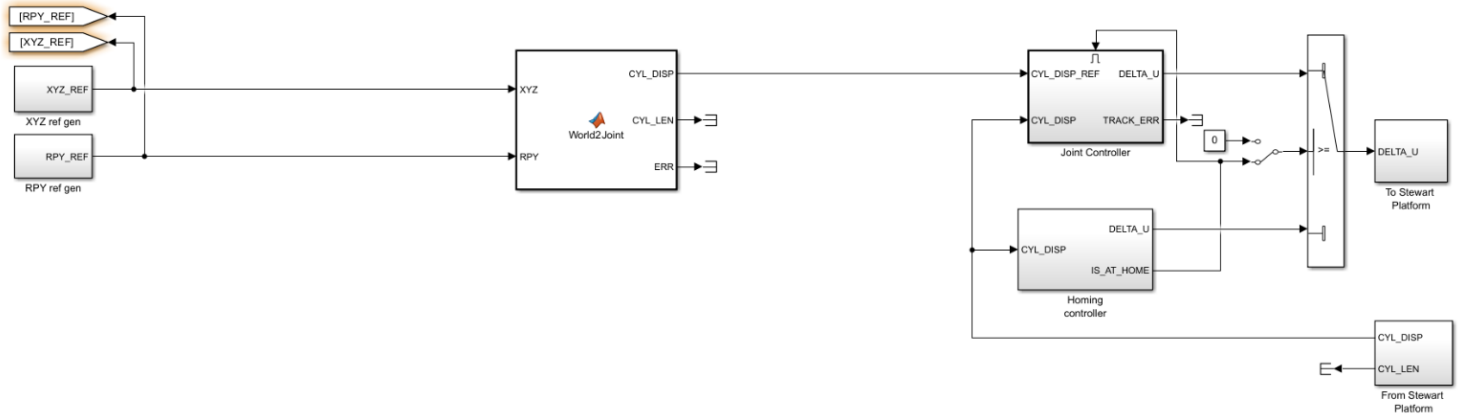
Ricezione (MEAS)

Trasmissione (CMD — apre il ciclo)

Ordine per ciclo: Simulink invia CMD → STM32 applica al DAC → STM32 risponde con MEAS. Un solo scambio Tx e uno Rx per ciclo.

Modello Simulink di controllo

Vista d'insieme dell'anello di controllo real-time della piattaforma Stewart



Generatore riferimenti

XYZ_REF + RPY_REF → posa desiderata

Cinematica inversa

World2Joint → CYL_DISP di ogni cilindro

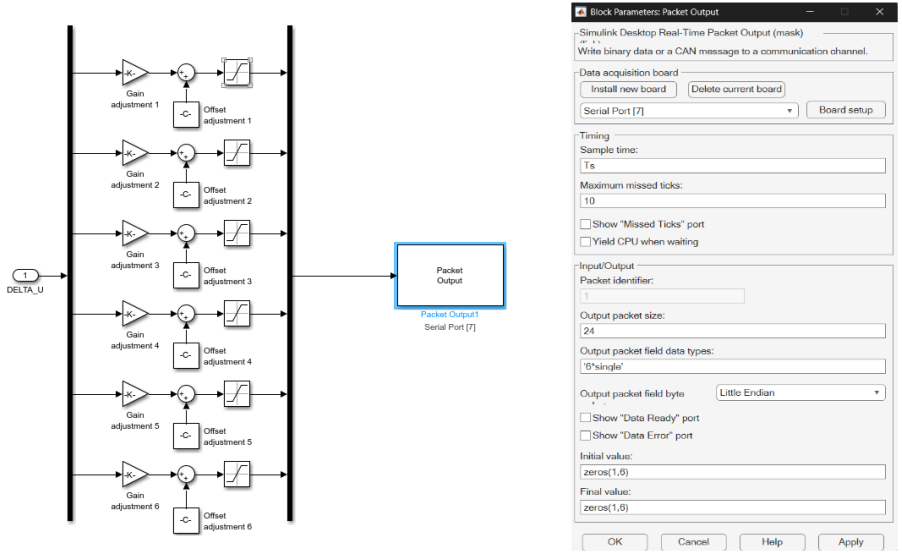
Loop di controllo

Joint / Homing controller → DELTA_U

CONTRIBUTO Modello di controllo sviluppato dal dott. Antonello; il nostro intervento si concentra sui sottosistemi "To/From Stewart Platform" dove abbiamo sostituito l'I/O analogico con i blocchi Packet Output / Packet Input.

"To Stewart Platform" · sottosistema Tx

Il DELTA_U calcolato viene condizionato e trasmesso al firmware STM32



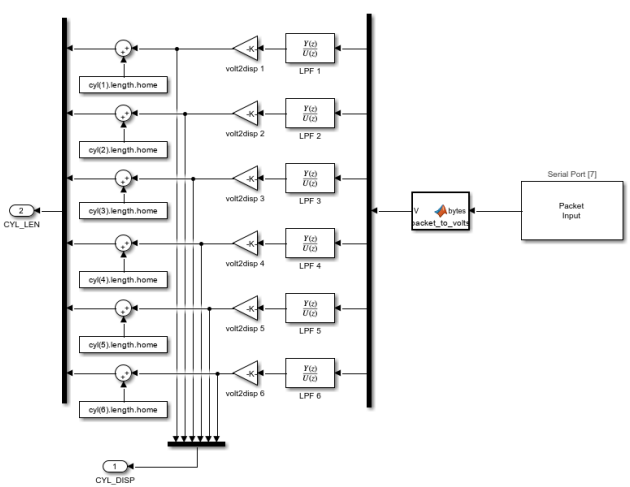
Catena di uscita

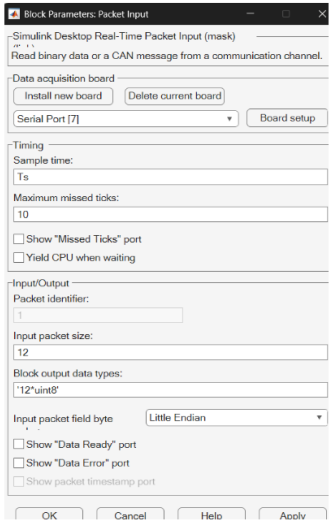
- DELTA_U (float) viene splittato nei 6 canali cilindro.
- Gain adjustment e Offset adjustment
- Saturazione: comando confinato in [-3, +3] V.
- Packet Output: invia via seriale a 115 200 bps allo STM32.

MODIFICA NOSTRA Packet Output configurato con packet size = 24 byte, formato «6*single", Little Endian; sample time = Ts, coerente con il ciclo di controllo.

"From Stewart Platform" · sottosistema Rx

Le misure ADC dallo STM32 vengono ricostruite in posizioni CYL_DISP





Catena di ingresso

- Packet Input: legge 12 byte uint8 dallo STM32 a ogni Ts.
- packet_to_volts: ricostruisce i 6 float (tensioni ADC).
- LPF i: filtro passa-basso digitale per attenuare il rumore.
- volt2disp i: LUT tensione → posizione, per cilindro.
- Sottrazione di cyl(i).length.home: coordinate relative allo zero.
- Output: CYL_DISP (posizioni) e CYL_LEN (allungamenti).

MODIFICA NOSTRA Packet Input configurato con packet size = 12 byte, formato "12*uint8", Little Endian; la volt2disp usa la LUT ottenuta dalla calibrazione.

07

Problemi incontrati

Cosa non ha funzionato e come è stato risolto

- Handshake DTR/DSR con FT232RL
- USART3 e Alternate Function
- Sincronizzazione Simulink / STM32
- Sequenza I^2C corretta del DAC7578
- UART_Reset e recupero errori

FT232RL e handshake DTR / DSR

Un vicolo cieco che ha guidato il design

Problema

- Prima ipotesi: UART7 con modulo esterno FT232RL.
- Idea: usare DTR/DSR per capire quando MATLAB apriva la seriale.
- Il modulo FT232 disponibile esponeva solo DTR, non DSR.
- Handshake hardware DTR/DSR impossibile.
- Abilitare DTR/DSR nei Packet In/Out avrebbe potuto bloccare la comunicazione.

Soluzione adottata

- Rimosso completamente FT232RL dal setup.
- Passaggio a USART3 (Virtual COM ST-LINK).
- Nessun controllo di flusso hardware.
- Sincronizzazione implicita: Simulink apre il ciclo col pacchetto CMD.
- Simulink resta master del tempo → sistema più robusto e semplice.

Sequenza I²C del DAC7578

Le uscite non cambiano insieme

Sintomo

- Ogni canale si aggiornava subito dopo la sua scrittura.
- I 6 canali cambiavano uno dopo l'altro, con circa 200 µs di ritardo.
- Gli attuatori partivano in modo asimmetrico ad ogni ciclo.
- Causa: usato il comando "scrivi e applica subito" per ogni canale.

Fix

- Sequenza corretta in due passi:

1) Scrivi i 6 valori nei buffer interni del DAC (uno per canale),

2) Un singolo comando broadcast li applica tutti insieme.

- Le 6 uscite analogiche cambiano nello stesso istante.
- Verificato con oscilloscopio: sfasamento praticamente nullo.

UART_Reset bloccante

Reset pulito della UART dopo timeout / errore

Sintomo

- Dopo un timeout della UART, la periferica resta "sporca".
- Rimangono byte nel registro e flag di errore attivi (ORE, RXNE).
- Quei byte "vecchi" verrebbero letti come parte del pacchetto successivo.
- Risultato: Simulink e STM32 restano fuori sincro per sempre.

Fix

- Ferma la ricezione in corso con HAL_UART_AbortReceive().
- Svuota il buffer leggendo i byte residui finché non è vuoto.
- Azzera i flag di errore (overrun) con la macro dedicata.
- Richiamato ad ogni errore: il ciclo riparte sempre da uno stato pulito.

Procedura di calibrazione

Costruzione della look-up table tensione → posizione per ciascun cilindro

Obiettivo

- Determinare la LUT che lega la tensione di comando (0 – 3.3 V) alla corsa reale del cilindro (-10 ÷ +10 cm).
- La curva è non lineare e diversa per ciascuno dei 6 cilindri: ogni cilindro richiede la sua calibrazione.
- Ogni cilindro viene comandato singolarmente (CYL_ID = 0 → 5) mentre gli altri restano al valore neutro.
- Al termine di ogni prova si ottiene un vettore LUT_uV[8] di tensioni corrispondenti alle posizioni [0, 1, ..., 7] cm (estensione utilizzata nella calibrazione), la procedura va poi ripetuta per estensioni negative.

Macchina a stati (STATE)

STATE = 0 — **Ricerca del target**

Il segnale DELTA_U corregge la tensione per portare CYL_DISP verso il valore obiettivo.

STATE = 1 — **Punto raggiunto**

Il cilindro è sul target ± tolleranza: si registra u nella LUT e si passa al punto successivo.

STATE = 2 — **Cilindro completato**

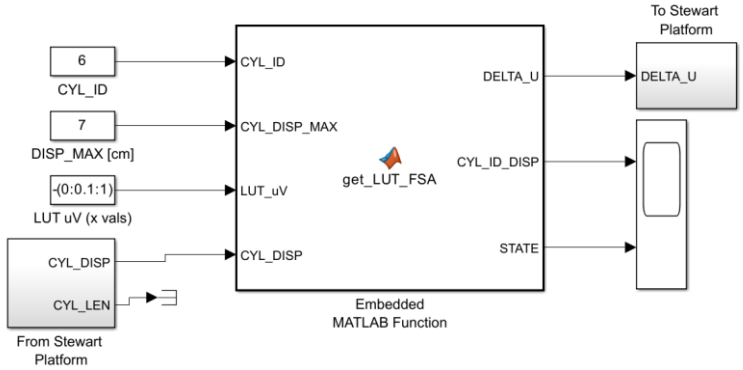
Tutti gli 8 punti della LUT sono acquisiti: fine calibrazione per il cilindro corrente.

NOTA Modello di calibrazione fornito dal Prof. Oboe; il nostro contributo è stato adattare i blocchi Packet Input / Output al nuovo protocollo digitale con STM32.

Calibrazione · modello e risultati

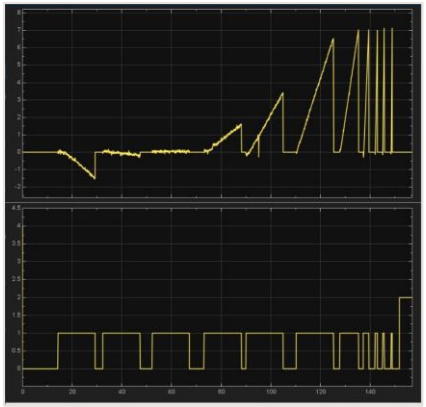
Il modello Simulink in esecuzione real-time e i segnali osservati

Modello Simulink di calibrazione

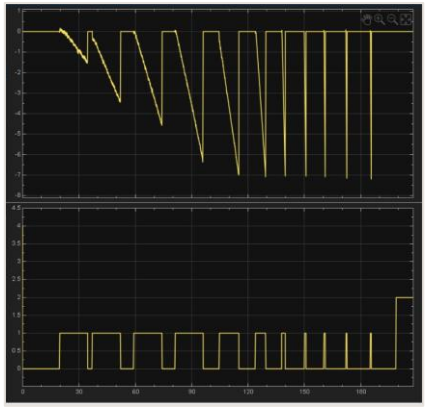


Ciclo Desktop Real-Time: get_LUT_FSA riceve CYL_DISP dallo STM32 e produce DELTA_U.

Traccia degli scope (DELTA_U e STATE)



Calibrazione pistone 4 per allungamento positivo

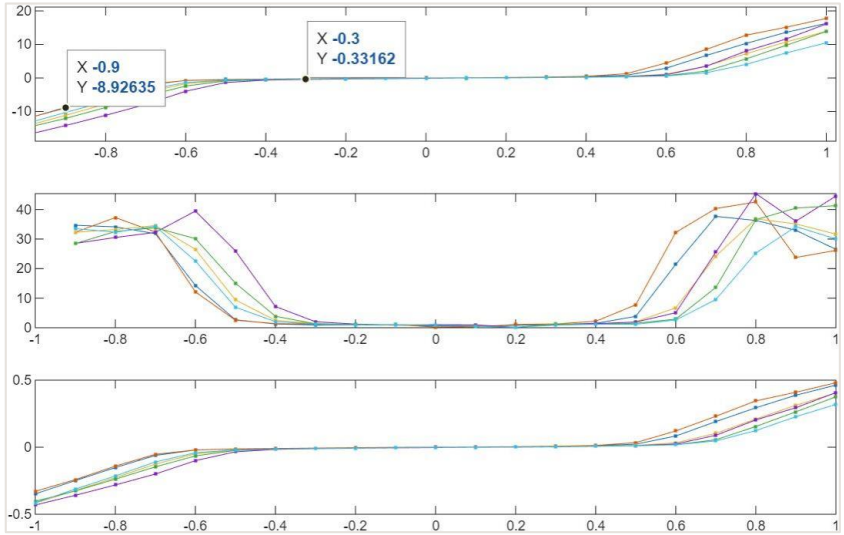


Calibrazione pistone 4 per allungamento negativo

Zona morta degli attuatori

Caratterizzazione sperimentale della non-linearità intorno allo zero

Risposta degli attuatori a comandi crescenti



Lettura dei grafici ottenuti

Asse x: comune ai tre grafici, riporta la tensione di test usata per comandare le valvole, compresa tra -1 e +1V.

Grafico 1: velocità del pistone in mm/s.

Grafico 2: guadagno locale (sensibilità) calcolato come variazione di velocità su variazione di tensione (derivata puntuale).

Grafico 3: Tensione di comando equivalente, ottenuta normalizzando la velocità misurata rispetto al guadagno massimo del tratto lineare .

COMPENSAZIONE La zona morta viene compensata via software sommando al comando un offset di segno concorde a DELTA_U, così da "saltare" l'intervallo insensibile e ottenere una risposta lineare anche per piccoli comandi.

08

Obiettivi raggiunti

Riepilogo dei risultati ottenuti

- Comunicazione digitale stabile Simulink ⇄ STM32
- Aggiornamento sincrono dei 6 attuatori
- Feedback ADC + DMA affidabile
- Calibrazione e compensazione della zona morta
- Setup low-cost, portabile e riproducibile

Risultati raggiunti

Cosa funziona oggi

Comunicazione digitale stabile

USART3 a 115200 bps con sincronizzazione implicita: il CMD apre il ciclo, la MEAS lo chiude.

Aggiornamento sincrono degli attuatori

6 canali DAC allineati grazie alla sequenza write-buffer + broadcast-update del DAC7578.

Feedback continuo via ADC + DMA

Snapshot coerenti in adc_latest, gestione corretta della cache L1.

Setup semplificato

Un solo cavo USB, niente FT232, niente Analog Input Quanser.

Modello Simulink portabile

Catena Packet In/Out + Byte Pack/Unpack riutilizzabile su altri progetti.

Firmware resiliente

Timeout su UART, saturazione dei comandi, modalità sicura del DAC allo startup.

Piattaforma in movimento

Video dimostrativo del sistema completo in funzione

Descrizione del video

La piattaforma ha 6 gradi di libertà:

- *3 di Traslazione (X,Y,Z)*
- *3 di Rotazione*

Grazie per l'attenzione

Domande?

UNIVERSITÀ DI PADOVA · 1222

Scuola di Ingegneria · Dipartimento di Tecnica e Gestione dei Sistemi Industriali (DTG)

A.A. 2025 / 2026