Introduzione alla scheda driver servo bus / adattatore servo bus XIAO

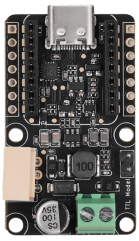
wiki.seeedstudio.com/bus_servo_driver_board

May 27, 2025

Questa wiki tratta due prodotti correlati: la **scheda driver per servobus** e l' **adattatore per servobus XIAO** .

- La **scheda driver per servocomandi bus** non include un microcontrollore XIAO ESP32- **C3** integrato, né è dotata di un contenitore stampato in 3D. È progettata per funzionare come scheda di interfaccia per servocomandi bus generica, consentendo di collegare e controllare i servocomandi tramite un controller esterno a scelta.
- L' **adattatore servo bus XIAO** , invece, **include** l'XIAO ESP32-C3 come controller principale ed è dotato di un case stampato in 3D. Con questa versione, è possibile controllare direttamente i servo bus utilizzando l'XIAO integrato, rendendolo una soluzione più integrata e pronta all'uso per progetti di robotica.

Per i dettagli sulla configurazione e l'utilizzo di entrambi i prodotti, consultare il resto della guida.

Scheda driver servobus	Adattatore servo bus XIAO
	
Prendine uno adesso 🖱️	Prendine uno adesso 🖱️

Introduzione

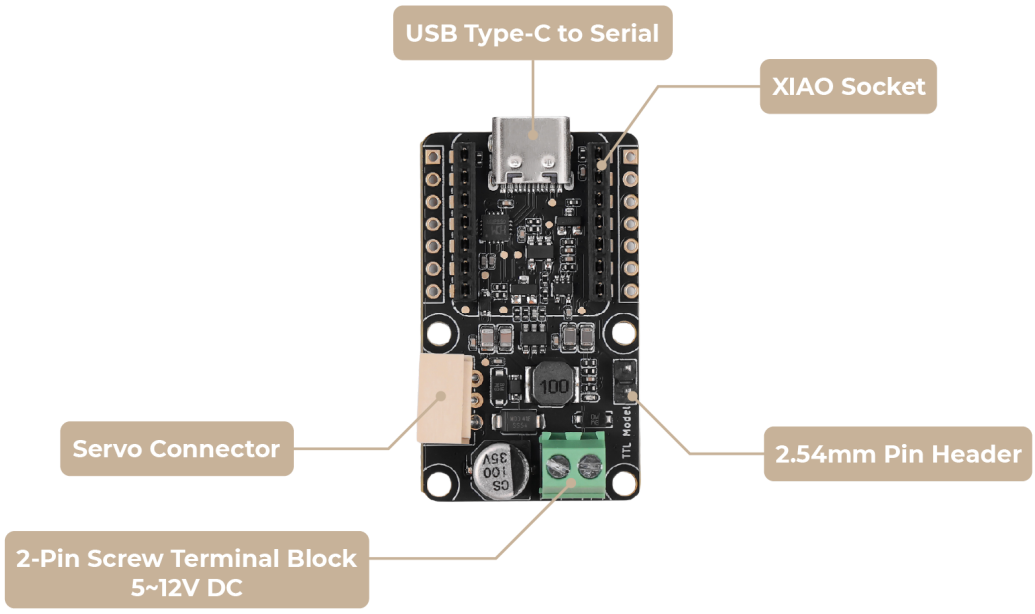
La scheda driver per servobus/adattatore per servobus XIAO è una soluzione hardware compatta e potente di Seeed Studio, progettata per pilotare servobus seriali per progetti di robotica e automazione. Grazie al supporto della comunicazione UART, consente un controllo e un feedback precisi da più servocomandi della serie ST/SC, inclusa la serie Feetech SCS (vedere il sito web ufficiale della serie Feetech SCS/STS/TTL). Questo la rende ideale per applicazioni come bracci robotici, esapodi, robot umanoidi e robot su ruote che richiedono feedback sull'angolo del servo e sul carico.

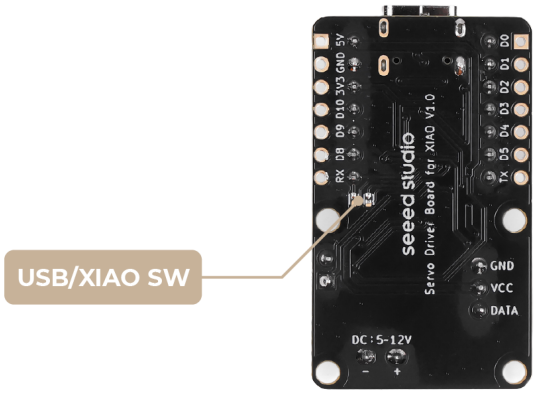
Questa guida si concentra sulla configurazione hardware, sulle connessioni fisiche, sulle specifiche chiave e **sulle impostazioni critiche dei jumper** per aiutare gli utenti a integrare efficacemente la scheda nei loro progetti.

⚠Avviso di sicurezza
Scollegare sempre l'alimentazione prima di collegare o scollegare i servocomandi o i cavi. Assicurarsi che la tensione di ingresso corrisponda ai requisiti del servo per evitare danni.

Panoramica dell'hardware

- Scheda driver servobus
- Adattatore servo bus XIAO





La scheda driver del servobus presenta diversi punti di connessione chiave:

Ingresso:

> **DC IN (5,5 * 2,1 mm):** questo è l'ingresso di alimentazione per la scheda e i servocomandi collegati. Collegare qui un alimentatore da 5~12 V. *È fondamentale che la tensione di questo alimentatore corrisponda ai requisiti di tensione dei servocomandi.* Ad esempio, i servocomandi della serie ST funzionano in genere a 9 V, mentre i servocomandi della serie SC potrebbero richiedere 12 V.

Produzione:

> **Interfaccia servo:** questa porta dedicata è quella a cui collegare i servocomandi della serie ST/SC. Assicurarsi che il connettore sia correttamente allineato.

Interfaccia di controllo:

> **UART (RX/TX):** questi pin forniscono la comunicazione seriale per il controllo dei servocomandi. Il metodo di connessione e le impostazioni dei jumper dipendono dal dispositivo host. Vedi sotto per i dettagli.

Per iniziare

Selezione della modalità operativa della scheda driver (solo per Bus Servo Driver Board)

mancia
Per l'adattatore servo bus XIAO, non è necessario modificare alcun circuito per utilizzare l'XIAO ESP32-C3 incluso per controllare i servo; è possibile saltare direttamente questa parte.

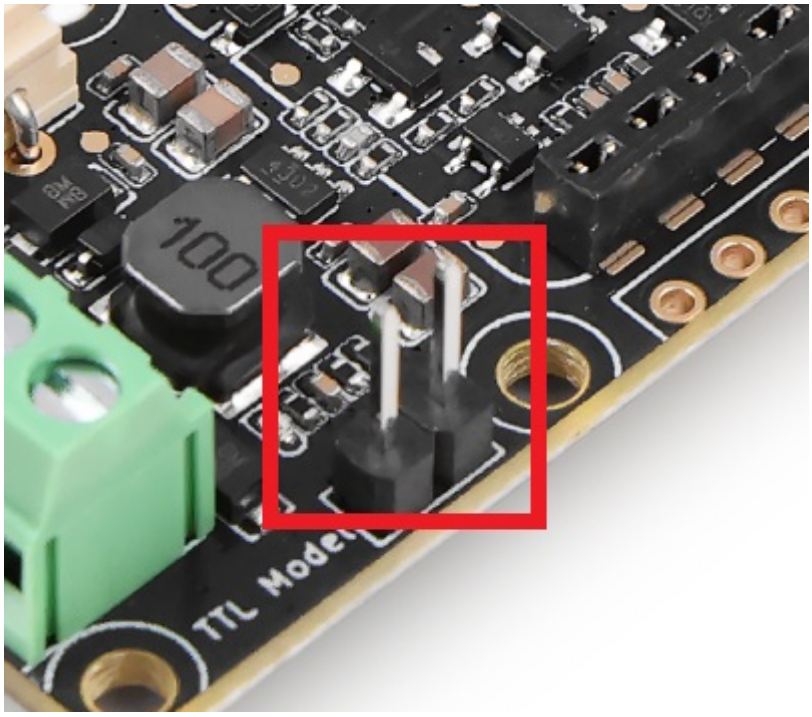
La scheda driver servobus offre due metodi di connessione principali: connessione UART diretta e connessione USB tramite un adattatore USB-UART. *La corretta impostazione dei jumper è essenziale per il corretto funzionamento.*

Connessione UART (per MCU, XIAO, ESP32, ecc.)

Questo metodo viene utilizzato quando ci si collega direttamente ai pin UART di un microcontrollore (MCU) come un ESP32, Arduino, Seeed Studio XIAO o un computer a scheda singola.

- **Cablaggio:**
  - Collegare il RXpin sulla scheda driver al TXpin (D7) sul dispositivo host.
  - Collegare il TXpin sulla scheda driver al RXpin (D6) sul dispositivo host.
  - Per dispositivi come Seeed Studio XIAO, è possibile collegare direttamente lo XIAO agli header forniti, garantendo il corretto allineamento dei pin. Ciò elimina la necessità di cavi Dupont separati per la connessione UART.
- **Impostazione del ponticello (critica):**

  Utilizzare un ponticello da 2,54 mm per cortocircuitare il pin a 2 pin sulla parte anteriore della scheda. (È cortocircuitato per impostazione predefinita)



- **Alimentazione dell'host:** il dispositivo host (ad esempio, Raspberry Pi Zero, ESP32, XIAO) richiederà un alimentatore separato.

Connessione USB

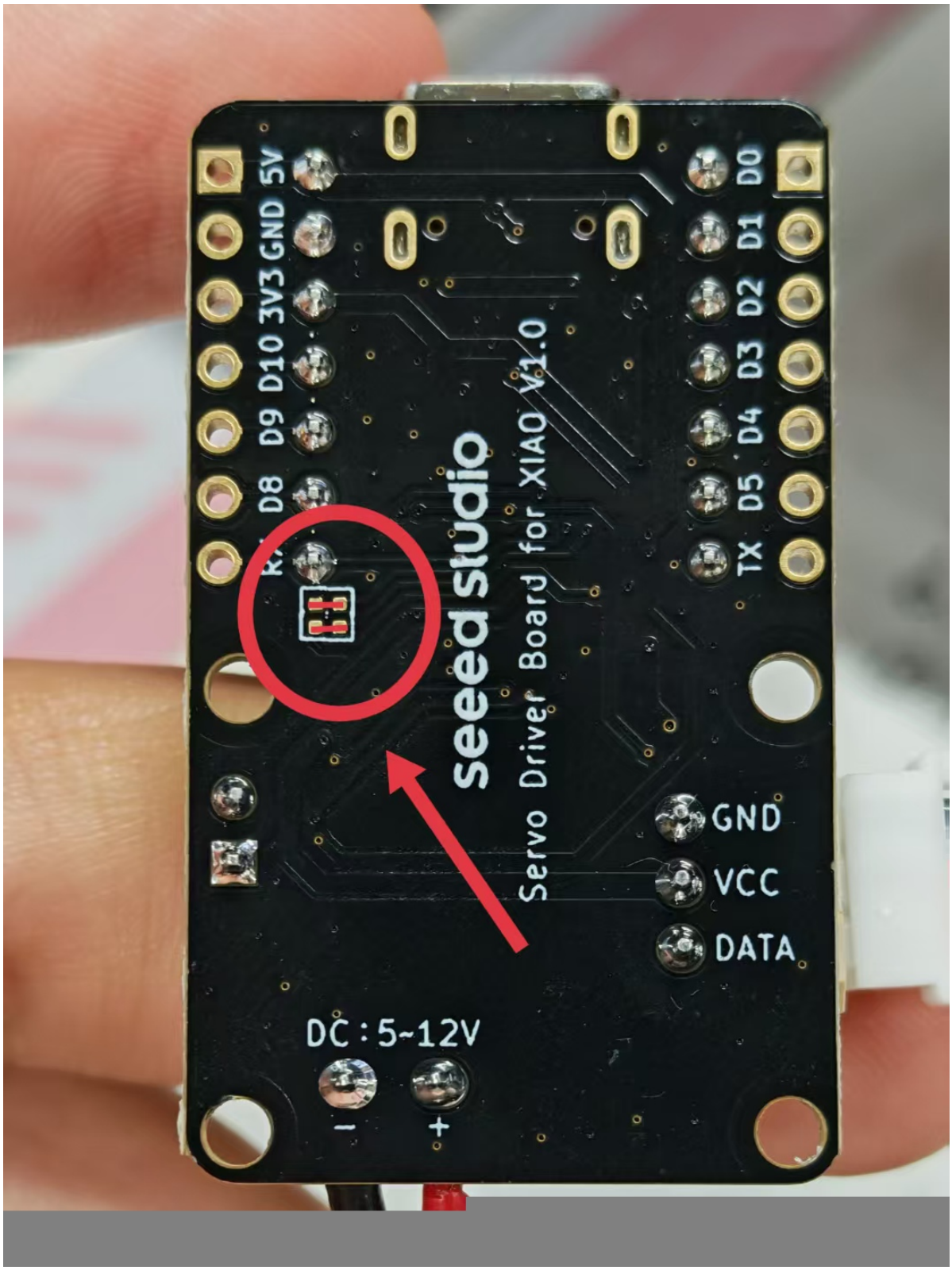
Questo metodo viene utilizzato quando ci si collega a un computer o a un computer a scheda singola dotato di porta USB (ad esempio, un PC o un Raspberry Pi 4B). È sufficiente collegare la scheda di controllo al computer tramite un cavo USB.

- **Cablaggio:**

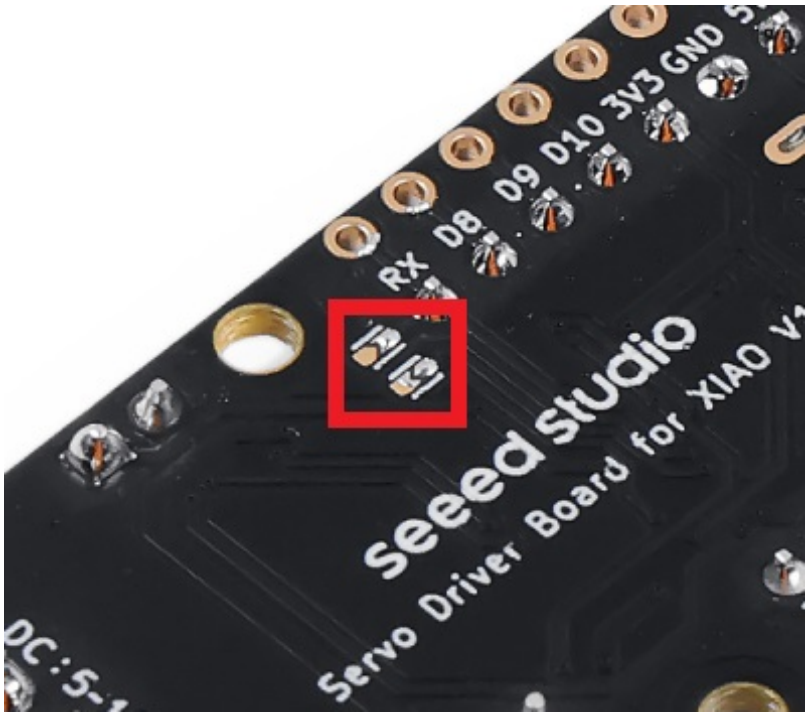
  Basta collegare la scheda di controllo al computer tramite un cavo USB.

- **Impostazione del ponticello (critica):**

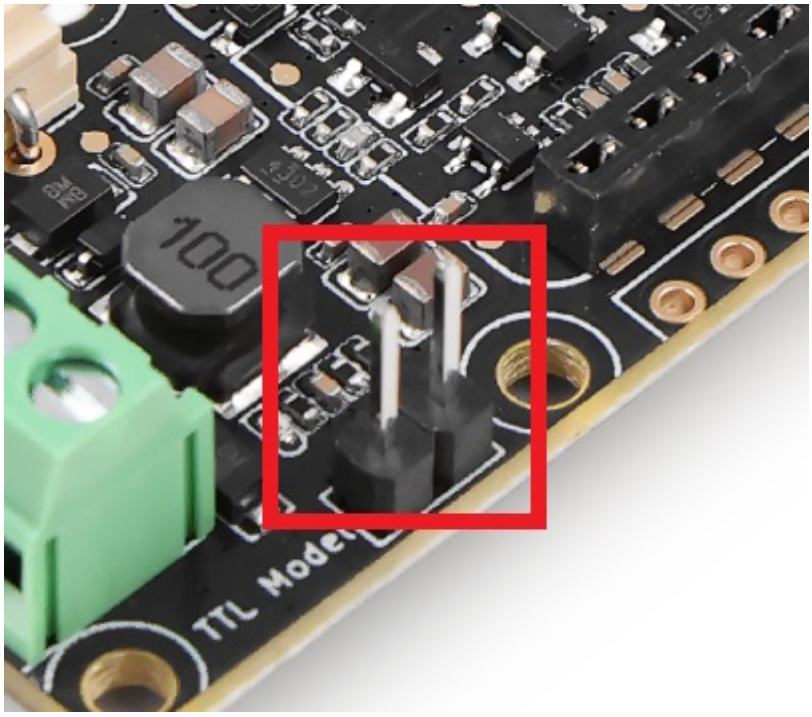
  Fase 1. Individuare il ponticello di saldatura sul retro della scheda. **Per la comunicazione USB, è necessario assicurarsi che i due pad siano collegati (tra di essi è presente un ponticello di saldatura).**
  Cuscinetti posteriori per la versione 1:



  Cuscinetti posteriori per la versione 2:



Passaggio 2. Utilizzare un ponticello da 2,54 mm per cortocircuitare il pin a 2 pin sulla parte anteriore della scheda. (È cortocircuitato per impostazione predefinita)



Componenti necessari (prima di iniziare)

Prima di collegare qualsiasi cosa, assicurati di avere quanto segue:

- **Scheda driver servo bus / Adattatore servo bus XIAO**
- **Servocomandi bus serie ST/SC compatibili** : vedere il sito Web ufficiale della serie SCS/STS/TTL di Feetech .

- **Alimentazione 5~12V:** una batteria o un adattatore di alimentazione. *La tensione deve corrispondere alle specifiche del servo.*
- **Dispositivo host:**
  - **Per UART diretto:** un dispositivo compatibile con UART come Raspberry Pi, Arduino, ESP32 o Seeed Studio XIAO.
  - **Per USB:** un computer (PC, Mac, Linux) o un computer a scheda singola come un Raspberry Pi 4B, *più* un adattatore da USB a UART.

ⓘnota
Per l'adattatore servo bus XIAO, XIAO ESP32-C3 è integrato, quindi non è necessario preparare un dispositivo host.

> **Cavi/adattatori di collegamento:** cavi jumper (cavi Dupont) se si utilizza UART diretto (tranne quando si utilizza XIAO con connessione diretta tramite header). Un adattatore da USB a UART se si utilizza il metodo di connessione USB.

⚠Attenzione
Se si utilizzano servocomandi della serie SC, verificare che l'alimentatore corrisponda ai requisiti di tensione. L'etichetta di ingresso CC della scheda è progettata appositamente per i servocomandi della serie ST, ma supporta anche le tensioni della serie SC.
Impostazioni errate dei jumper impediranno la comunicazione con la scheda driver.

Controllo dei servocomandi tramite USB

Questa sezione descrive come controllare più servocomandi bus tramite la scheda driver servo bus utilizzando una connessione USB.

Panoramica dei principi

La scheda driver per servocomandi bus funziona ricevendo comandi seriali (UART) dal dispositivo host (come un PC, un Raspberry Pi o un microcontrollore) tramite USB. Questi comandi vengono quindi inoltrati ai servocomandi bus collegati. Inviando i comandi del protocollo seriale appropriato, è possibile controllare individualmente la posizione, la velocità e altri parametri di ciascun servocomando.

La scheda stessa non interpreta né genera autonomamente i segnali di controllo dei servocomandi; funge invece da ponte trasparente tra l'host e i servocomandi. Ciò significa che sei responsabile dell'invio dei pacchetti di comando corretti in base al protocollo di comunicazione del tuo servocomando.

Esempio di riferimento

Per un esempio pratico su come inviare comandi ai servocomandi bus Feetech (serie ST/SC/STS/TTL), puoi fare riferimento al seguente esempio Python:
lerobot/common/robot_devices/motors/feetech.py su GitHub

Questo esempio mostra come costruire e inviare pacchetti seriali per controllare i servocomandi Feetech. È possibile adattare il codice alla propria piattaforma host e al proprio linguaggio di programmazione, se necessario.

> **Nota:**
>
> - Il formato e il protocollo specifici del comando possono variare a seconda del modello del servo.
> - Per conoscere il protocollo seriale e la struttura dei comandi corretti, consultare la documentazione ufficiale del servo.
> - Sarà necessario scrivere o adattare un programma driver che soddisfi i requisiti del tuo servo.

Per maggiori dettagli sul protocollo della serie SCS/STS/TTL di Feetech, consultare la documentazione ufficiale di Feetech .

Controllo dei servocomandi tramite XIAO

Successivamente, descriviamo come inviare segnali per controllare il movimento del servo tramite XIAO e come utilizzare la libreria.

Panoramica della libreria Arduino

mancia
Se è la prima volta che utilizzi Arduino, ti consigliamo vivamente di consultare la guida introduttiva ad Arduino .

Scarica la libreria -

Funzione

Prima di iniziare a sviluppare uno sketch, diamo un'occhiata alle funzioni disponibili nella libreria.

- `SMS_STS(uint8_t id)`—— Crea un oggetto servo con l'ID specificato.
  Parametri: `uint8_t id`(ID servo)
  Output: nessuno

- `void WritePos(uint8_t id, int16_t Position, uint16_t Time, uint16_t Speed)`—— Imposta la posizione target, il tempo e la velocità per il servo.
  Parametri: `uint8_t id`, `int16_t Position`, `uint16_t Time`, `uint16_t Speed`
  Output: nessuno

- `void RegWritePos(uint8_t id, int16_t Position, uint16_t Time, uint16_t Speed)`—— Imposta la posizione target, il tempo e la velocità per il servo, ma esegui in seguito con il comando Azione.
  Parametri: `uint8_t id`, `int16_t Position`, `uint16_t Time`, `uint16_t Speed`
  Output: nessuno

- `void RegWriteAction()`—— Esegue tutti i comandi RegWritePos registrati.
  Parametri: nessuno
  Output: nessuno

- `void WriteSpe(uint8_t id, int16_t Speed)`—— Imposta la velocità di rotazione del servo.
  Parametri: `uint8_t id`, `int16_t Speed`
  Uscita: nessuna

- `void WritePosEx(uint8_t id, int16_t Position, uint16_t Time, uint16_t Speed, uint8_t ACC)`—— Imposta posizione, tempo, velocità e accelerazione.
  Parametri: `uint8_t id`, `int16_t Position`, `uint16_t Time`, `uint16_t Speed`, `uint8_t ACC`
  Output: nessuno

- `void RegWritePosEx(uint8_t id, int16_t Position, uint16_t Time, uint16_t Speed, uint8_t ACC)`—— Registra posizione, tempo, velocità e accelerazione, esegui in seguito.
  Parametri: `uint8_t id`, `int16_t Position`, `uint16_t Time`, `uint16_t Speed`, `uint8_t ACC`
  Output: nessuno

- `void RegWriteActionEx()`—— Esegue tutti i comandi RegWritePosEx registrati.
  Parametri: nessuno
  Output: nessuno

- `int16_t ReadPos(uint8_t id)`—— Legge la posizione attuale del servo.
  Parametri: `uint8_t id`
  Uscita: `int16_t`(posizione)

- `int16_t ReadSpeed(uint8_t id)`—— Legge la velocità attuale del servo.
  Parametri: `uint8_t id`
  Uscita: `int16_t`(velocità)

- `int16_t ReadLoad(uint8_t id)`—— Legge il carico attuale del servo.
  Parametri: `uint8_t id`
  Uscita: `int16_t`(carico)

- `int16_t ReadVoltage(uint8_t id)`—— Legge la tensione attuale del servo.
  Parametri: `uint8_t id`
  Uscita: `int16_t`(tensione)

- `int16_t ReadTemper(uint8_t id)`—— Legge la temperatura attuale del servo.
  Parametri: `uint8_t id`
  Uscita: `int16_t`(temperatura)

- `int16_t ReadMove(uint8_t id)`—— Controlla se il servo si muove.
  Parametri: `uint8_t id`
  Uscita: `int16_t`(1: in movimento, 0: fermo)

- `int16_t ReadCurrent(uint8_t id)`—— Legge la corrente (corrente elettrica) del servo.
  Parametri: `uint8_t id`
  Uscita: `int16_t`(corrente)

- `void SetID(uint8_t id, uint8_t newid)`—— Imposta un nuovo ID per il servo.
  Parametri: `uint8_t id`, `uint8_t newid`
  Output: nessuno

- `void Load(uint8_t id)`—— Abilita la coppia del servo.
  Parametri: `uint8_t id`
  Uscita: nessuna

- `void Unload(uint8_t id)`—— Disabilita la coppia del servo.
  Parametri: `uint8_t id`
  Uscita: nessuna

- `int16_t ReadTorque(uint8_t id)`—— Legge lo stato di coppia del servo.
  Parametri: `uint8_t id`
  Uscita: `int16_t`(1: abilitato, 0: disabilitato)

- `void LEDAlarm(uint8_t id, uint8_t enable)`—— Imposta lo stato di allarme del LED.
  Parametri: `uint8_t id`, `uint8_t enable`
  Uscita: nessuna

- `void Reset(uint8_t id)`—— Ripristina il servo alle impostazioni di fabbrica.
  Parametri: `uint8_t id`
  Uscita: nessuna

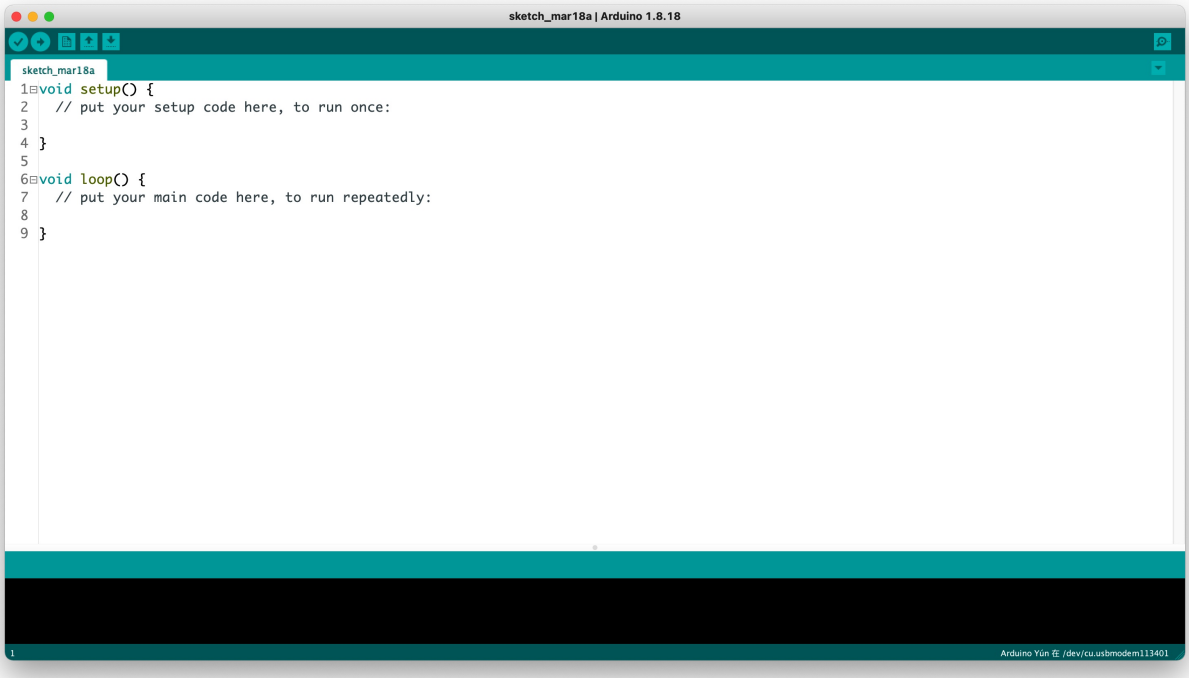
- `void LockEprom(uint8_t id)`—— Blocca la EEPROM del servo.
  Parametri: `uint8_t id`
  Uscita: nessuna

- `void UnlockEprom(uint8_t id)`—— Sblocca la EEPROM del servo.
  Parametri: `uint8_t id`
  Uscita: nessuna

Esempio XIAO

Ora che abbiamo installato la nostra libreria e ne abbiamo compreso le funzioni di base, eseguiamo alcuni esempi per il nostro 产品名称 per vedere come si comporta.

Passaggio 1. Avviare l'applicazione Arduino.

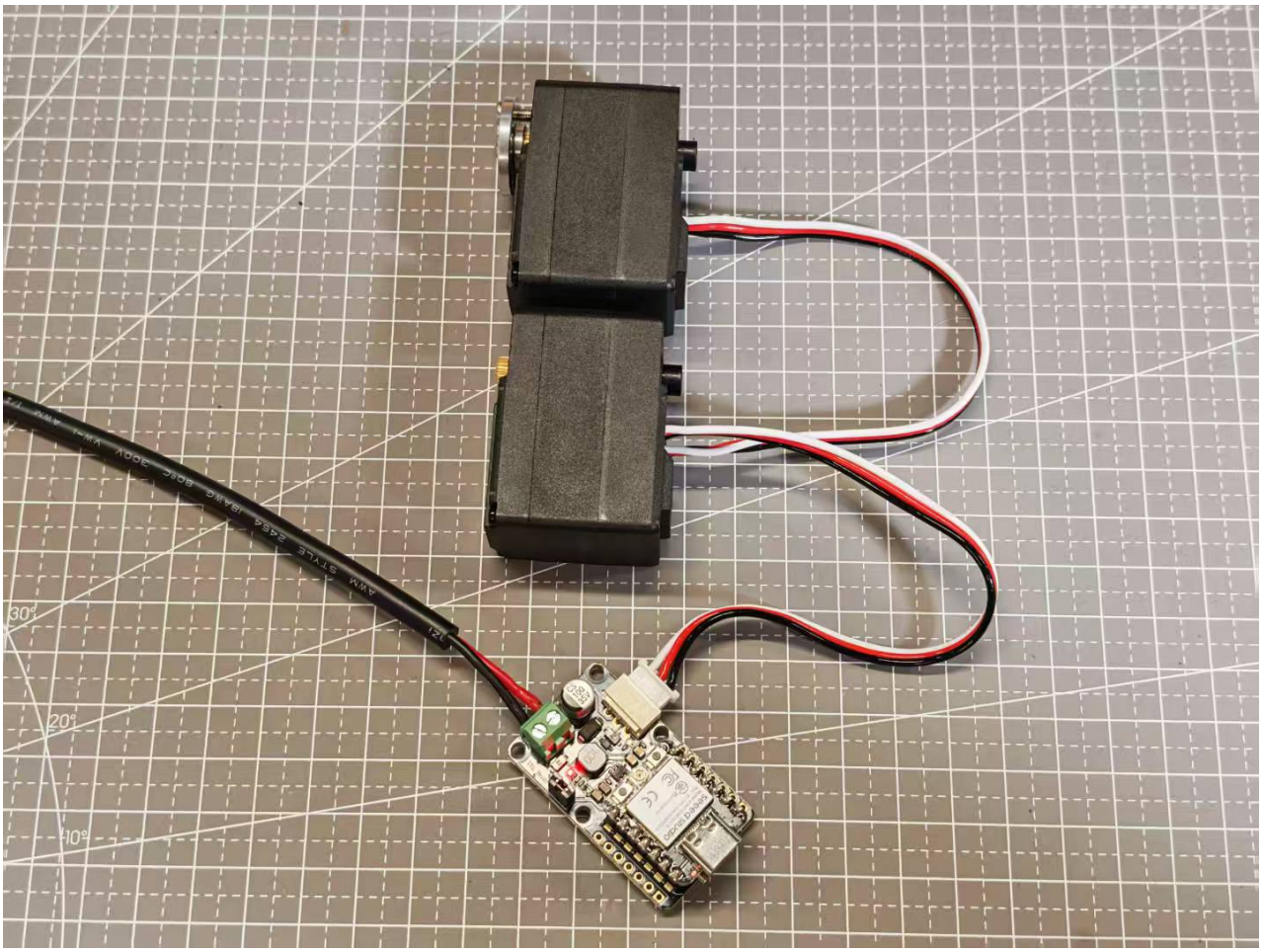


[Scarica Arduino IDE]

Passaggio 2. Seleziona il modello della tua scheda di sviluppo e aggiungilo all'IDE di Arduino.

> Per utilizzare **Seeed Studio XIAO ESP32-C3** per le routine successive, fare riferimento a **questo tutorial** per completare l'aggiunta.

Fase 3. Completare il cablaggio come mostrato. Se è necessario collegare più servi, è possibile utilizzare i cavi forniti con i servi per completare il collegamento.


AWM STYLE 2464 18AWG 80°C 300V VW-1
30°
20°
10°

```
#include<SCServo.h>
// Define the correct serial port for your target board
#ifdefined(CONFIG_IDF_TARGET_ESP32C3)||defined(CONFIG_IDF_TARGET_ESP32C6)||defined
(CONFIG_IDF_TARGET_ESP32S3)
#defineCOMSerialSerial0
#else
#defineCOMSerialSerial1
#endif
// Define RX/TX pins for the servo bus (for reference)
// Note: On ESP32, pins are usually specified in COMSerial.begin().
// For example: COMSerial.begin(1000000, SERIAL_8N1, S_RXD, S_TXD);
// If your board uses the default pins for Serial1, no extra specification is
needed.
#defineS_RXDD7
#defineS_TXDD6
#defineSERVO_NUM2// Number of servos
SMS_STS st;// Servo control object
// --- Servo Configuration ---
byte ID[SERVO_NUM]={1,2};// IDs of the servos
u16 Speed[SERVO_NUM]={1500,1500};// Set a medium speed for the servos
byte ACC[SERVO_NUM]={50,50};// Set a medium acceleration for the servos
s16 Pos[SERVO_NUM]={2048,2048};// Servo position array, initialized to the
midpoint (2048)
voidsetup()
{
// Start the main serial port for debugging and receiving commands
  Serial.begin(115200);
// Wait a moment for the Serial Monitor to connect
delay(2000);
  Serial.println("--- Servo Control Program Start ---");
// Start the serial port for controlling the servos
  COMSerial.begin(1000000, SERIAL_8N1);
  st.pSerial =&COMSerial;// Associate the control object with the serial port
  Serial.println("Checking servo connection status...");
for(int i =0; i < SERVO_NUM; i++){
if(st.Ping(ID[i])!=-1){
      Serial.print("Servo with ID ");
      Serial.print(ID[i]);
      Serial.println(" is connected.");
}else{
      Serial.print("Error: Servo with ID ");
      Serial.print(ID[i]);
      Serial.println(" is not responding!");
 // --- Power-on Self-Test ---
// This section makes the servos move automatically on power-up to confirm they
are working correctly.
  Serial.println("\nExecuting power-on self-test movement...");
// 1. Move to position 1024
  Serial.println("Moving to position 1024...");
for(int i=0; i<SERVO_NUM; i++){
    Pos[i]=1024;
}
  st.SyncWritePosEx(ID, SERVO_NUM, Pos, Speed, ACC);
delay(2000);// Wait for the movement to complete
// 2. Move to position 3072
  Serial.println("Moving to position 3072...");
```

```
for(int i=0; i<SERVO_NUM; i++){
    Pos[i]=3072;
}
  st.SyncWritePosEx(ID, SERVO_NUM, Pos, Speed, ACC);
delay(2000);// Wait for the movement to complete
// 3. Return to center position (2048) to prepare for user commands
  Serial.println("Returning to center position (2048)...");
for(int i=0; i<SERVO_NUM; i++){
    Pos[i]=2048;
}
  st.SyncWritePosEx(ID, SERVO_NUM, Pos, Speed, ACC);
delay(1500);
  Serial.println("\n--- Initialization Complete ---");
  Serial.println("Enter 'j' to decrease the angle, or 'k' to increase it.");
  Serial.println("-----------------------------------");
 voidloop()
{
// Check if the user has sent a command via the Serial Monitor
if(Serial.available()){
    String input = Serial.readString();
    input.trim();// Remove extra spaces or newlines
bool shouldMove =false;// Flag to indicate if a valid command was received
if(input.startsWith("j")){
      Serial.println("Received command: 'j'. Decreasing angle.");
for(int i =0; i < SERVO_NUM; i++){
        Pos[i]-=512;// Move a small step for easy observation
if(Pos[i]<0){
          Pos[i]=0;// Prevent going below the minimum range
       shouldMove =true;
}elseif(input.startsWith("k")){
      Serial.println("Received command: 'k'. Increasing angle.");
for(int i =0; i < SERVO_NUM; i++){
        Pos[i]+=512;// Move a small step
if(Pos[i]>4095){
          Pos[i]=4095;// Prevent going above the maximum range
       shouldMove =true;
}else{
      Serial.print("Unknown command: '");
      Serial.print(input);
      Serial.println("'. Please enter 'j' or 'k'.");
 // If a valid command was received, send the new positions to the servos
if(shouldMove){
      Serial.print("Moving servos to new positions: [");
for(int i =0; i < SERVO_NUM; i++){
        Serial.print(Pos[i]);
if(i < SERVO_NUM -1) Serial.print(", ");
}
      Serial.println("]");
      st.SyncWritePosEx(ID, SERVO_NUM, Pos, Speed, ACC);
```

Questo esempio mostra come controllare più servocomandi bus della serie Feetech SCS utilizzando XIAO e la libreria SCServo. Il codice inizializza due servocomandi, li calibra e consente all'utente di regolarne interattivamente la posizione tramite comandi seriali. Quando si inviano "j" o "k" tramite il monitor seriale, il codice diminuirà o aumenterà

rispettivamente l'angolo di tutti i servocomandi collegati. La posizione corrente di ciascun servocomando viene tracciata e aggiornata di conseguenza, e le nuove posizioni vengono inviate ai servocomandi tramite la `SyncWritePosEx`funzione.

Come personalizzare il tuo progetto:

- **Numero di servocomandi** : modifica il valore `Servo_Num`e aggiorna gli array ID, Velocità, ACC e Pos in modo che corrispondano al numero e agli ID dei tuoi servocomandi. ID servocomandi: modifica l'array ID in modo che corrisponda agli ID dei tuoi servocomandi collegati.
- **Velocità e accelerazione** : regola le matrici Velocità e ACC per impostare velocità e accelerazioni diverse per ciascun servo.
- **Pin seriali** : se si utilizzano pin diversi per UART, aggiornare le definizioni S_RXD e S_TXD.
- **Logica di movimento** : è possibile modificare la logica nella `loop()`funzione per implementare comportamenti più complessi o specifici del progetto, come la risposta a diversi comandi seriali, l'aggiunta di feedback del sensore o l'integrazione con altro hardware.
- **Posizione iniziale** : imposta i valori iniziali nella `Pos`matrice per definire le posizioni iniziali dei tuoi servocomandi.

Domande frequenti

mancia
Si consiglia di leggere attentamente queste FAQ prima di iniziare il progetto. Rispondono a domande comuni e potenziali problemi.

▶ Cosa succede se la tensione di alimentazione non è adatta al mio servo?
▶ Posso collegare più servi contemporaneamente?

Risorse

[PDF] Schema della scheda driver del servobus

Supporto tecnico e discussione

Grazie per aver scelto i nostri prodotti! Siamo qui per offrirti diversi tipi di supporto per garantire che la tua esperienza con i nostri prodotti sia la più fluida possibile. Offriamo diversi canali di comunicazione per soddisfare diverse preferenze ed esigenze.