Guide Configuration Servos SO-ARM 101

Phase 2 : Attribution des IDs et Tests des Servomoteurs

Développé par : Service Ecoles Médias (SEM)

📋 Prérequis

- Phase 1 complétée (LeRobot installé)
- Environnement `lerobot` activé
- 2 adaptateurs USB-Serial (Waveshare ou Feetech)
- Alimentations :
  - Kit Standard : 2×5V 3A
  - Kit Pro : 1×5V 3A + 1×12V 2A
- 12 servos Feetech STS3215
- Câbles 3-pins fournis

> **Note :** Cette phase utilise exclusivement le script SEM développé pour le Service Ecoles Médias, optimisé pour l'éducation et la formation.

🎯 Vue d'ensemble de la Phase 2

Cette phase consiste à :

1. **Attribuer un ID unique** à chaque servo (1 à 6)
2. **Tester le mouvement** de chaque servo
3. **Centrer** chaque servo en position 2048
4. **Monter** les servos sur la structure
5. **Reconfigurer si nécessaire** après montage

🔧 Étape 1 : Préparation de l'environnement

Activation de l'environnement LeRobot

```
# Activer l'environnement conda
conda activate lerobot

# Se placer dans le dossier de travail
cd ~/lerobot

# Vérifier que l'environnement est actif
# Vous devez voir (lerobot) au début de votre ligne de commande
```

Configuration matérielle

Composant	Leader	Follower
Adaptateur USB	1 adaptateur dédié	1 adaptateur dédié
Alimentation	5V 3A (toujours)	5V ou 12V selon kit
Servos	3 types (ratios différents)	Tous identiques (1:345)

⚠ **Important :** Ne JAMAIS connecter plusieurs servos non configurés simultanément ! Configurez-les un par un.

📝 Étape 2 : Création du script SEM_so101_config_servo.py

Ce script pédagogique permet de configurer chaque servo individuellement avec détection automatique du port USB.

Création du script

Copiez et collez ce bloc complet dans votre terminal :

```
cat > SEM_so101_config_servo.py << 'EOF'
#!/usr/bin/env python3
"""
Script SEM_so101_config_servo.py
Service Ecoles Médias - Configuration des servos SO-ARM 101

Ce script permet de :
1. Détecter automatiquement le port USB
2. Attribuer un ID à un servo
3. Tester ses mouvements
4. Le centrer en position 2048
"""

import time
import sys
import os

# Ajout du chemin LeRobot pour importer les bibliothèques
sys.path.append('/home/prof/lerobot')

def detect_usb_port():
    """
    Détection automatique du port USB
    Teste les ports courants et retourne le premier disponible
    """
    print("🔍 Détection automatique du port USB...")

    # Liste des ports USB possibles
    ports_possibles = ['/dev/ttyACM0', '/dev/ttyACM1', '/dev/ttyACM2',
                       '/dev/ttyUSB0', '/dev/ttyUSB1']

    for port in ports_possibles:
        if os.path.exists(port):
            print(f"✅ Port trouvé : {port}")
            # Donner les permissions nécessaires
            try:
                with open(port, 'r'):
                    pass
            except PermissionError:
                print(f"⚠️  Permissions insuffisantes, correction...")
                os.system(f"sudo chmod 666 {port}")
                print(f"✅ Permissions accordées pour {port}")
            return port

    print("❌ Aucun port USB détecté")
```

```
    print("Vérifiez que l'adaptateur USB est bien branché")
    return None

# Affichage du titre
print("""
╔═══════════════════════════════════════════════════════╗
║     SEM - SERVICE ECOLES MÉDIAS                       ║
║     Configuration & Test Servo SO-ARM 101             ║
║     Avec détection automatique                        ║
╚═══════════════════════════════════════════════════════╝
""")

# Détection du port USB
PORT = detect_usb_port()
if not PORT:
    print("\n✗ Impossible de continuer sans port USB")
    print("Connectez l'adaptateur USB et relancez le script")
    exit(1)

print(f"\n🔌 Port USB actif : {PORT}")
print("⚡ Alimentation : Vérifiez que l'alimentation 5V est branchée")
print("⚡ Servo : Branchez UN SEUL servo à configurer\n")

# Menu de sélection du servo
print("=== SÉLECTION DU SERVO ===")
print("  1 - Base (rotation horizontale)")
print("  2 - Épaule (monte/descend le bras)")
print("  3 - Coude (plie/déplie)")
print("  4 - Poignet flexion (haut/bas)")
print("  5 - Poignet rotation (gauche/droite)")
print("  6 - Pince/Poignée")
print("  D - [Détecter] un servo déjà configuré")
print("")

choix = input("Entrez le numéro du servo (1-6) ou D pour détecter : ").strip().upper()

# Import des bibliothèques Dynamixel après la sélection
from dynamixel_sdk import *

if choix == 'D':
    # Mode détection - trouve l'ID d'un servo déjà configuré
    print("\n=== MODE DÉTECTION ===")
    portHandler = PortHandler(PORT)
    packetHandler = PacketHandler(1.0)  # Protocol version 1.0 pour STS3215

    if portHandler.openPort() and portHandler.setBaudRate(1000000):
        print("Recherche de servos configurés...")
        found = False

        # Recherche sur les IDs 1 à 25
        for servo_id in range(1, 26):
            result, _, _ = packetHandler.ping(portHandler, servo_id)
            if result == 0:  # 0 = succès
                pos, _, _ = packetHandler.read2ByteTxRx(portHandler, servo_id, 56)
                print(f"✓ Servo trouvé : ID={servo_id}, Position={pos}")
                found = True

        if not found:
            print("✗ Aucun servo configuré détecté")
```

```
            print("   Le servo n'a pas encore d'ID ou n'est pas alimenté")

        portHandler.closePort()
    else:
        print("✗ Erreur de connexion au port")
    exit()

# Vérification de la validité du choix
try:
    SERVO_ID = int(choix)
    if SERVO_ID < 1 or SERVO_ID > 6:
        print("✗ Numéro invalide! Choisissez entre 1 et 6")
        exit(1)
except:
    print("✗ Entrée invalide!")
    exit(1)

# Dictionnaire des noms de servos pour l'affichage
servo_names = {
    1: "Base",
    2: "Épaule",
    3: "Coude",
    4: "Poignet flexion",
    5: "Poignet rotation",
    6: "Pince/Poignée"
}

# PARTIE 1 : Configuration du servo avec l'ID choisi
print(f"\n=== CONFIGURATION SERVO {SERVO_ID} ({servo_names[SERVO_ID]}) ===")
print(f"1. Attribution de l'ID {SERVO_ID} et mise au centre (position 2048)...")

# Construction de la commande LeRobot pour configurer le servo
cmd = f"python lerobot/scripts/configure_motor.py --port {PORT} --brand feetech --model
result = os.system(cmd)

if result != 0:
    print("✗ Erreur lors de la configuration!")
    print("Vérifiez que :")
    print("  - Le servo est bien connecté")
    print("  - L'alimentation est active")
    print("  - Un seul servo est branché")
    exit(1)

print("✓ Configuration terminée, attente 2 secondes...")
time.sleep(2)

# PARTIE 2 : Test de mouvement du servo
print(f"\n2. Test de mouvement du servo {SERVO_ID}")

# Configuration de la communication
BAUDRATE = 1000000
MOTOR_ID = SERVO_ID

# Adresses des registres pour STS3215
ADDR_TORQUE_ENABLE = 40      # Adresse pour activer/désactiver le couple
ADDR_GOAL_POSITION = 42      # Adresse pour la position cible
ADDR_PRESENT_POSITION = 56   # Adresse pour lire la position actuelle
PROTOCOL_VERSION = 1.0        # Version du protocole Dynamixel
```

```
# Définition des positions de test
POS_MIN = 1024    # Position -90 degrés
POS_CENTER = 2048 # Position 0 degrés (centre)
POS_MAX = 3072    # Position +90 degrés

# Initialisation de la communication
portHandler = PortHandler(PORT)
packetHandler = PacketHandler(PROTOCOL_VERSION)

# Ouverture du port
if not portHandler.openPort():
    print(f"✗ Impossible d'ouvrir le port {PORT}")
    print("Le port est peut-être utilisé par un autre programme")
    exit(1)

# Configuration de la vitesse de communication
if not portHandler.setBaudRate(BAUDRATE):
    print(f"✗ Impossible de configurer le baudrate {BAUDRATE}")
    exit(1)

print(f"✓ Connecté au servo ID {MOTOR_ID} sur {PORT}")

# Activation du couple moteur
packetHandler.write1ByteTxRx(portHandler, MOTOR_ID, ADDR_TORQUE_ENABLE, 1)

try:
    # Lecture de la position actuelle
    present_pos, _, _ = packetHandler.read2ByteTxRx(portHandler, MOTOR_ID, ADDR_PRESENT_
    print(f"📍 Position actuelle: {present_pos}")

    print(f"\n=== Test de mouvement SERVO {SERVO_ID} ({servo_names[SERVO_ID]}) ===")

    # Test 1 : Position MIN
    print("\n1️⃣ MIN (1024) - Rotation gauche/bas...")
    packetHandler.write2ByteTxRx(portHandler, MOTOR_ID, ADDR_GOAL_POSITION, POS_MIN)
    time.sleep(2)  # Attendre que le mouvement soit complet

    # Test 2 : Position MAX
    print("2️⃣ MAX (3072) - Rotation droite/haut...")
    packetHandler.write2ByteTxRx(portHandler, MOTOR_ID, ADDR_GOAL_POSITION, POS_MAX)
    time.sleep(2)

    # Test 3 : Retour au CENTRE
    print("3️⃣ CENTRE (2048) - Position centrale...")
    packetHandler.write2ByteTxRx(portHandler, MOTOR_ID, ADDR_GOAL_POSITION, POS_CENTER)
    time.sleep(2)

    # Lecture de la position finale
    final_pos, _, _ = packetHandler.read2ByteTxRx(portHandler, MOTOR_ID, ADDR_PRESENT_P

    print(f"\n✓ SUCCÈS ! Servo {SERVO_ID} configuré et centré")
    print(f"📍 Position finale : {final_pos}")

    # Instructions pour le montage
    print("\n" + "="*50)
    print("🔧 INSTRUCTIONS POUR LE MONTAGE :")
    print("="*50)

    if SERVO_ID == 6:
```

```
        print("➜ Servo 6 (Pince) : ")
        print("  1. Montez le palonnier avec la pince OUVERTE")
        print("  2. La position 2048 = pince ouverte (position de repos)")
    else:
        print(f"➜ Servo {SERVO_ID} ({servo_names[SERVO_ID]}) : ")
        print("  1. Le servo est maintenant centré (2048)")
        print("  2. Montez le palonnier en position droite/alignée")
        print("  3. Fixez le servo sur la structure")

    print("\n💡 CONSEIL :")
    print("Après montage, si la position n'est plus correcte :")
    print("→ Relancez ce script avec le même numéro pour recentrer")

except KeyboardInterrupt:
    print("\n\n⚠ Interruption par l'utilisateur")

except Exception as e:
    print(f"\n✖ Erreur : {e}")

finally:
    # Fermeture propre de la connexion
    portHandler.closePort()
    print("\n🔌 Port fermé")

print("\n✨ Configuration terminée avec succès!")
EOF

# Rendre le script exécutable
chmod +x SEM_so101_config_servo.py

echo "✅ Script SEM_so101_config_servo.py créé avec succès!"
```

📡 Étape 3 : Procédure de configuration

A. Configuration du bras LEADER

> **Rappel Leader :** Les servos du Leader ont des ratios différents :
>
> - Servos 1 et 3 : Ratio 1:191 (marquage C044)
> - Servo 2 : Ratio 1:345 (marquage C001)
> - Servos 4, 5, 6 : Ratio 1:147 (marquage C046)

Procédure pour chaque servo :

1. **Préparation matérielle :**
   - Brancher l'adaptateur USB du Leader
   - Connecter l'alimentation 5V 3A
   - Ne brancher qu'UN SEUL servo à la fois
2. **Lancement du script :**

   ```
   python SEM_so101_config_servo.py
   ```

3. **Configuration :**
   - Choisir le numéro du servo (1 à 6)
   - Observer le test de mouvement (MIN → MAX → CENTRE)
   - Vérifier que le servo finit bien à la position 2048
4. **Montage sur la structure :**
   - Monter le palonnier en position alignée
   - Fixer le servo sur le bras
   - Si la position bouge pendant le montage, relancer le script

B. Configuration du bras FOLLOWER

> **Rappel Follower :** Tous les servos du Follower sont identiques (ratio 1:345)

Répéter la même procédure avec l'adaptateur USB du Follower.

> 💡 **Astuce :** Après avoir monté chaque servo, il est normal que la position centrale soit perdue. N'hésitez pas à relancer le script pour recentrer le servo après montage. C'est pourquoi nous configurons AVANT le montage (pour avoir l'ID) puis APRÈS si nécessaire (pour recentrer).

🔍 Étape 4 : Vérification et dépannage

Utilisation du mode Détection

Pour identifier un servo déjà configuré :

```
python SEM_so101_config_servo.py
# Choisir 'D' pour détecter
```

Le script affichera l'ID et la position actuelle du servo connecté.

Tableau de dépannage

Problème	Causes possibles	Solutions
Port USB non détecté	• Adaptateur non branché • Mauvais port USB • Permissions insuffisantes	• Vérifier le branchement • Essayer un autre port USB • Lancer : `sudo chmod 666 /dev/ttyACM*`
Servo ne bouge pas	• Alimentation non connectée • Câble 3-pins mal branché • Servo défectueux	• Vérifier l'alimentation (LED allumée) • Reconnecter le câble 3-pins • Tester avec un autre servo
Position incorrecte après montage	• Normal - le montage fait bouger le servo • Palonnier mal positionné	• Relancer le script pour recentrer • Démonter et remonter le palonnier
Plusieurs servos détectés	• Plusieurs servos connectés en chaîne	• Débrancher tous sauf un • Configurer un par un
ID déjà utilisé	• Servo déjà configuré • Mauvais servo branché	• Utiliser mode 'D' pour identifier • Brancher le bon servo

📝 Notes importantes

⚠ **Règles essentielles :**

1. **Un servo à la fois :** Ne JAMAIS connecter plusieurs servos non configurés
2. **Position 2048 :** Toujours configurer avant montage
3. **Reconfiguration :** Normal et recommandé après montage
4. **Alimentation :**
   - Leader : TOUJOURS 5V
   - Follower : 5V (Standard) ou 12V (Pro)
5. **Ordre :** Configurer → Tester → Monter → (Reconfigurer si besoin)

🔄 Commandes rapides de référence

```
# Activer l'environnement
conda activate lerobot
cd ~/lerobot

# Configurer un servo
python SEM_so101_config_servo.py

# Détecter un servo existant
python SEM_so101_config_servo.py
# Puis choisir 'D'

# Permissions USB si nécessaire
sudo chmod 666 /dev/ttyACM*
```