

Machine Learning LeRobot avec SO-ARM101

Installation et prérequis

Prérequis pour l'entraînement et l'exécution d'un modèle d'IA :

- Une carte graphique NVidia et une installation de Cuda
- L'entrainement avec 100 épisodes et 100 000 steps a mis 12H sur une RTX 2080 Super de 2019
 - Il n'aboutit pas au bout de plus de 24H sur une Quadro P620 (via WSL2 et Docker)
- L'exécution semble OK sur une Quadro P620

Installation sous Linux

- [Installer Miniconda pour Linux](#) : l'environnement de développement Python

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
# Vérifier que la clé SHA256 de Miniconda3-latest-Linux-x86_64.sh ici :
https://repo.anaconda.com/miniconda/ correspond à :
sha256sum ~/Miniconda3-latest-Linux-x86_64.sh
bash ~/Miniconda3-latest-Linux-x86_64.sh
source ~/.bashrc
```

- Créer et activer l'environnement Conda

```
conda create -y -n lerobot python=3.10
conda activate lerobot
git clone https://github.com/huggingface/lerobot.git ~/lerobot
conda install ffmpeg=7.1.1 -c conda-forge
cd ~/lerobot && pip install -e ".[feetech]"
```

- A chaque ouverture de Terminal l'environnement python conda est activé, voir au bas du

```
~/ .bashrc
```

- Pour éviter les conflits, on propose d'avoir un fichier `~/.bashrc_conda` pour conda et un `~/.bashrc_ros` pour ros

Astuces pour activer/désactiver l'environnement conda sans passer par la modification du `~/.bashrc` :

- Ne pas activer conda au démarrage : `conda config --set auto_activate_base false`
- Ne pas configurer le shell pour initialiser conda au démarrage : `conda init --reverse $SHELL`

Installation Windows

- Le compte utilisateur doit avoir les droits pour créer des raccourcis (liens symboliques) dans les sous-dossiers de `C:\Users\%USER%\lerobot\outputs\train\`
- Ils seront utilisés lors de l'entraînement pour créer un lien entre le dossier `last` et le dossier du dernier Checkpoint par ex. `100000`
 - Le plus sûr est de **travailler avec un compte administrateur**
 - Il faut peut-être aussi les droits dans le dossier `C:\Users\%USER%\cache\huggingface\lerobot\%HUGGINGFACE_USER`
- [Installer Miniconda pour Windows](#) : l'environnement de développement Python
- Ouvrir `Anaconda PowerShell Prompt`
- Créer et activer l'environnement Conda

```
conda create -y -n lerobot python=3.10
conda activate lerobot
git clone https://github.com/huggingface/lerobot.git ~/lerobot
```

- Installer Pytorch pour la version de Cuda installée sur votre système (testé avec une version Cuda 127 installée et la version cu128 de Pytorch) et autres dépendances nécessaires

```
cd ~/lerobot
# pip install av poetry-core
conda install ffmpeg=7.1.1 -c conda-forge
pip3 install --pre torch torchvision torchaudio --index-url
https://download.pytorch.org/whl/cu128
pip install -e ".[feetech]"
```

Récupérer les infos système pour déboguer l'installation si les scripts basculent sur le `cpu` :

- `python lerobot/scripts/display_sys_info.py`
- `python -m torch.utils.collect_env`
- `python -c "import torch; print(torch.cuda.is_available())" && nvcc -V`

- cf. <https://github.com/huggingface/lerobot/issues/928> > Here for additional information of my full installation.

Banc de Machine Learning LeRobot

Configurer les servomoteurs

La carte `FE-URT-1` fournie par Feetech n'est pas détectée sous Ubuntu à cause d'un conflit avec un paquet de brail. On le désinstalle :

```
sudo apt-get autoremove brlTTY
```

<https://askubuntu.com/questions/1321442/how-to-look-for-ch340-usb-drivers/1472246#1472246>

https://github.com/huggingface/lerobot/blob/main/examples/10_use_so100.md#c-configure-the-motors

- Brancher la carte
- Trouver l'interface USB sur laquelle est branchée la carte

```
python lerobot/scripts/find_motors_bus_port.py
```

- Sous Linux, par ex. `/dev/ttyACM0` ou `/dev/ttyUSB0`
- Sous Windows, par ex. `COM13` ou `COM14`
- Sous Linux, Changer les droits sur les interfaces USB

```
sudo chmod 666 /dev/ttyACM0  
sudo chmod 666 /dev/ttyACM1
```

- Ouvrir Codium > File > Open Folder > `admin_ros/lerobot`
- Modifier le fichier de config

```
gedit ~/lerobot/lerobot/common/robot_devices/robots/configs.py
```

- Chercher la config du So100 en ligne 436 `class So100RobotConfig(ManipulatorRobotConfig):`
- Remplacer `port="/dev/tty.usbmodem58760431091"`, pour le `leader_arms` (L446) et le `follower_arms` (L463) par le port découvert
- Brancher les servos un à un à la carte puis lancer le script d'initialisation, en incrémentant l'ID à chaque fois :

```
python lerobot/scripts/configure_motor.py \  
  --port /dev/tty.usbmodem58760432961 \  
  --brand feetech \  
  --model sts3215 \  
  --baudrate 1000000 \  
  --ID 1
```

- Au fur et à mesure les brancher en série et/ou noter l'ID sur le moteur
- Les servos sont bougés à leur position centrale et l'offset mis à 0

Ne plus bouger les servos jusqu'à l'assemblage

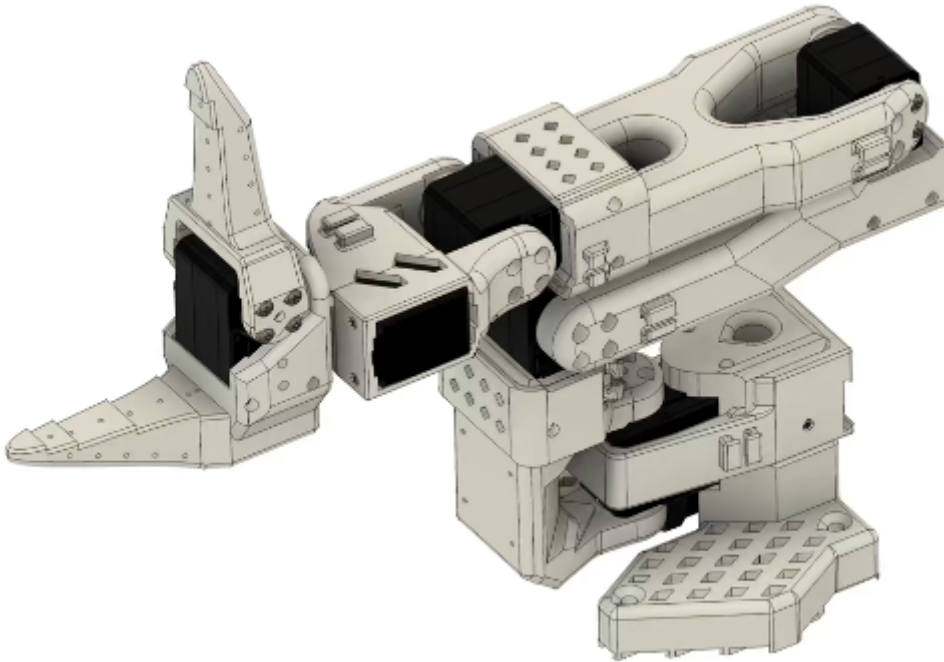
Construction et assemblage mécanique

Une version 101 est sortie en 05/2025. Le Leader est plus simple à assembler, et plus besoin de [démonter les servos pour enlever un engrenage et les rendre passifs](#). Il suffit d'acheter le kit de 6 servos avec 3 rapports de transmission différents :

- <https://github.com/TheRobotStudio/SO-ARM100?tab=readme-ov-file#getting-your-own-so101>
- https://www.alibaba.com/product-detail/6PCS-7-4V-STS3215-Servos-for_1601428584027.html?spm=a2747.product_manager.0.0.757c2c3cIU7uH3
- Imprimer la mâchoire statique intégrant le support de caméra : https://github.com/TheRobotStudio/SO-ARM100/blob/main/Optional/Wrist_Cam_Mount_32x32_UVC_Module/README.md
- Suivre le guide d'assemblage pour le SO101 : <https://huggingface.co/docs/lerobot/so101#step-by-step-assembly-instructions>
- Pour le SO100 : <https://huggingface.co/docs/lerobot/so100#step-by-step-assembly-instructions>

Astuces pour l'assemblage

- Mettre une vis sur l'arbre moteur et l'axe passif (à l'opposée de l'arbre moteur) quand il y a la place d'en mettre une (vérifier qu'il y aura la place après assemblage des éléments autour du moteur)
- Ne plus bouger les servos après leur initialisation qui les met à l'angle 0. Dans l'idéal, assembler les éléments de manière à ce que le robot soit en configuration initiale avec tous les moteurs à 0
- En pratique, on monte le robot dans la configuration ci-dessous. C'est l'étape de calibration qui permettra de définir un offset pour que le zéro des moteurs corresponde au



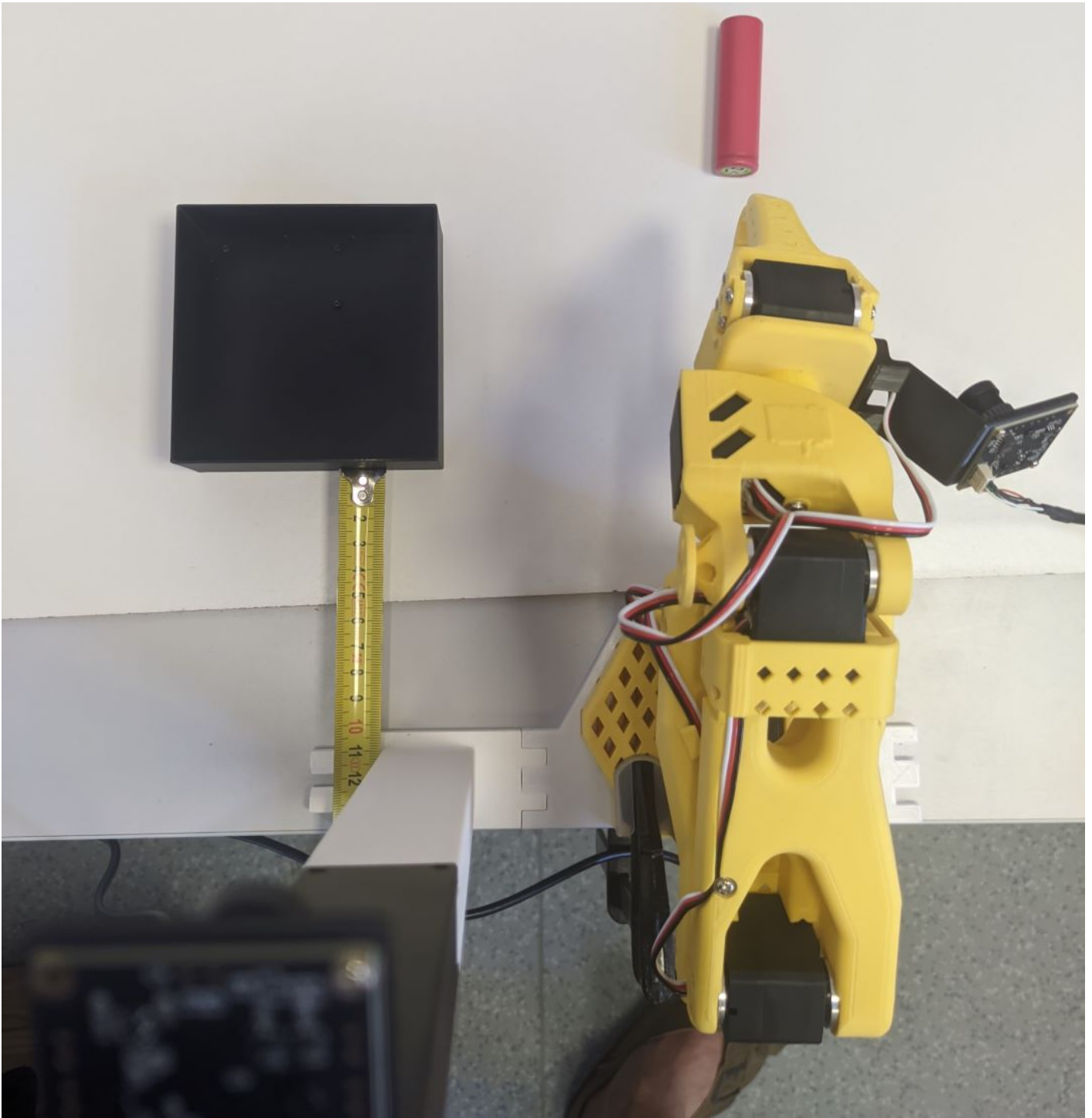
- Il est possible d'ajouter un offset dans la configuration des servomoteurs, par exemple via les scripts du projet LeRobot
- Attention si vous démarrez le robot sous ROS avant d'avoir lancé la calibration LeRobot qui fixe l'Offset dans les servomoteurs, vous risquez de casser le robot

Agencement des caméras et robots

Le nombre, le positionnement et la qualité des caméras sont importants pour la qualité du DataSet :

- Plusieurs setup sont proposés :
 - Caméras d'environnement : <https://github.com/TheRobotStudio/SO-ARM100?tab=readme-ov-file#2-overhead-camera-mount>
 - Caméras de poignet : <https://github.com/TheRobotStudio/SO-ARM100?tab=readme-ov-file#5-wristmount-cameras>
- Attention au champ de vision des caméras si vous prenez une de vos webcams
 - Il risque de ne pas être assez "fish eye"
 - Par exemple, la WebCam Logitech C270 (720p) a un champ trop étroit pour être intégrée au [module Overhead](#)

Au FabLab de IUT Haguenau



- On choisit de prendre deux caméras au format 32 x 32 , la version 1080p permet d'augmenter la qualité du DataSet
 - <https://www.amazon.com/innomaker-Computer-Raspberry-Support-Windows/dp/B0CNCSFQC1/132-7372155-9780230>
- Imprimer et assembler la mâchoire statique intégrant le support de caméra :
https://github.com/TheRobotStudio/SO-ARM100/blob/main/Optional/Wrist_Cam_Mount_32x32_UVC_Module/README.md
- Imprimer et assembler le support de robot et de caméra Overhead :
<https://github.com/TheRobotStudio/SO->

Calibration des caméras

<https://huggingface.co/docs/lerobot/cameras>

Utilisation de LeRobot

- Activer l'environnement conda lerobot

```
cd ~/lerobot
```

```
conda activate lerobot
```

```
python lerobot/scripts/find_motors_bus_port.py
```

```
nano lerobot/common/robot_devices/robots/configs.py
```

```
python lerobot/scripts/control_robot.py --robot.type=so101 --robot.cameras='{}' --  
control.type=teleoperate
```

Calibration robot et configuration caméras

```
python -m lerobot.calibrate --teleop.type=so101_leader --teleop.port=/dev/ttyACM0 --  
teleop.id=leader_arm_fan1
```

```
python -m lerobot.calibrate --robot.type=so101_follower --robot.port=/dev/ttyUSB0 --  
robot.id=follower_arm_fan1
```

Téléopération

```
python -m lerobot.teleoperate \  
  --robot.type=so101_follower \  
  --robot.port=/dev/ttyUSB0 \  
  --robot.id=follower_arm_fan1 \  
  --robot.cameras="{ top: {type: opencv, index_or_path: 2, width: 640, height: 480, fps:  
30}, follower: {type: opencv, index_or_path: 4, width: 640, height: 480, fps: 30}}" \  
  --teleop.type=so101_leader \  
  --teleop.port=/dev/ttyACM0 \  
  --teleop.id=leader_arm_fan1 \  
  --display_data=true
```

Rejouer dataset en local :

```
python -m lerobot.replay \  
  --robot.type=so101_follower \  
  --robot.port=/dev/ttyUSB0 \  
  --robot.id=follower_arm_fan1 \  
  --dataset.repo_id=gautz/18650-test1-10ep \  
  --dataset.episode=0 # choose the episode you want to replay
```

Machine Learning

Enregistrer dataset en local :

```
python -m lerobot.record \  
  --robot.type=so101_follower \  
  --robot.port=/dev/ttyUSB0 \  
  --robot.id=follower_arm_fan1 \  
  --robot.cameras="{ top: {type: opencv, index_or_path: 2, width: 640, height: 480, fps: 30}, follower: {type: opencv, index_or_path: 4, width: 640, height: 480, fps: 30}}" \  
  --teleop.type=so101_leader \  
  --teleop.port=/dev/ttyACM0 \  
  --teleop.id=leader_arm_fan1 \  
  --display_data=true \  
  --dataset.repo_id=gautz/18650-test1-10ep \  
  --dataset.episode_time_s=10 \  
  --dataset.reset_time_s=10 \  
  --dataset.num_episodes=10 \  
  --dataset.single_task="Pick red 18650 battery place black box" \  
  --dataset.push_to_hub=False
```

Entraîner en local avec le cpu

```
python lerobot/scripts/train.py \  
  --dataset.repo_id=gautz/18650-test1-10ep \  
  --policy.type=act \  
  --output_dir=outputs/train/act_so101_18650-test1-10ep \  
  --job_name=act_so101_18650-test1-10ep \  
  --policy.device=cpu # \  
  --wandb.enable=false # true
```


Entraîner en local avec le GPU

```
python lerobot/scripts/train.py \  
  --dataset.repo_id=gautz/18650-test1-10ep \  
  --policy.type=act \  
  --output_dir=outputs/train/act_so101_18650-test1-10ep \  
  --job_name=act_so101_18650-test1-10ep \  
  --policy.device=cuda \  
  --wandb.enable=false
```

Enregistrer un dataset d'évaluation d'un modèle à un checkpoint donné :

```
python -m lerobot.record \  
  --robot.type=so101_follower \  
  --robot.port=/dev/ttyUSB0 \  
  --robot.cameras="{ top: {type: opencv, index_or_path: 2, width: 800, height: 600, fps: 30}, follower: {type: opencv, index_or_path: 4, width: 800, height: 600, fps: 30}}" \  
  --robot.id=follower_arm_fan1 \  
  --display_data=false \  
  --dataset.repo_id=gautz/eval_act_18650-test2-100ep \  
  --dataset.single_task="Pick red 18650 battery place black box" \  
  --policy.path=outputs/train/act_so101_18650-test2-100ep/checkpoints/last/pretrained_model \  
  --dataset.push_to_hub=False
```

Visualiser et rejouer des DataSets (avant hardware refactor)

- Visualiser un DataSet

```
python lerobot/scripts/visualize_dataset.py --repo-id lerobot/pusht --root ./my_local_data_dir  
--local-files-only 1 --episode-index 0
```

```
python lerobot/scripts/visualize_dataset_html.py \  
  --repo-id cadene/act_so100_5_lego_test_080000 \  
  --local-files-only 1
```

- Rejouer un DataSet (ou une évaluation de modèle) sur le robot

```
python lerobot/scripts/control_robot.py \  
  --robot.type=so101 \  
  --control.type=replay
```

```
--control.fps=30 \  
--control.repo_id=cadene/act_so100_5_lego_test_080000 \  
--control.episode=0
```

- Rejouer la Policy `cadene/act_so100_5_lego_test_080000` du modèle ACT pour le SO-ARM101
 - En sauvegardant l'évaluation dans `outputs/eval/act_so100_5_lego_test_080000_haguenau`

```
python lerobot/scripts/control_robot.py \  
--robot.type=so101 \  
--control.type=record \  
--control.fps=30 \  
--control.single_task="Grasp a lego block and put it in the bin." \  
--control.repo_id=outputs/eval/act_so100_5_lego_test_080000_haguenau \  
--control.tags=['"tutorial"'] \  
--control.warmup_time_s=5 \  
--control.episode_time_s=30 \  
--control.reset_time_s=30 \  
--control.num_episodes=10 \  
--control.push_to_hub=false \  
--control.policy.path=cadene/act_so100_5_lego_test_080000
```

Sources

https://wiki.seeedstudio.com/lerobot_so100m/

Revision #26

Created 8 April 2025 20:29:23 by admin_idf

Updated 8 July 2025 12:10:01 by admin_idf