# Bras Robot - Arduino ROS2 IA


- Etat de l'art Bras robot low-cost
- SO-ARM100 - Robotique éducative
- ROS2 avec SO-ARM101
- IA robotique - Architectures pour l'apprentissage profond
- Machine Learning LeRobot avec SO-ARM101
- Pilotage des servomoteurs : TTL, RS232, RS485
- Transmission TTL et protocole RS485
- Introduction à Modbus
- Machine Learning LeRobot avec SO-ARM101 - Legacy
- Banc de machine learning avec SO-ARM101

# Etat de l'art Bras robot low-cost

## Modèles commerciaux fermés

### Niryo Ned 2

- 6DOF + Pince

https://niryo.com/fr/produit/bras-robotise-6-axes/

https://github.com/NiryoRobotics

### DAGU Six-servo Robot Arm

- 5DOF Manipulateur + 1DOF Pince
- 6 servos
  - 3x 13 kg.cm torque metal gear, 40.4 * 19.8 * 36 mm, 48g, 0.22s/60°

    - 1x 3.2 kg.cm, 39.5 x20.0x35.5mm, 41g, 0.27s/60°
    - 2x 2.3 kg.cm, 28 x14x29.8mm, 18g, 0.13/60°
- Carte de contrôle AREXX Intelligence Centre

https://seafile.unistra.fr/d/693101e6046d4819a3af/

https://arexx.com/product/robot-arm/

www.arexx.com.cn

# Modèles Open Source

https://github.com/AntoBrandi/Robotics-and-ROS-2-Learn-by-Doing-Manipulators

## Trossen Robotics ALOHA

### Stationary

https://docs.trossenrobotics.com/trossen_arm/main/specifications.html

https://docs.trossenrobotics.com/aloha_docs/2.0/specifications.html#aloha-stationary

https://docs.trossenrobotics.com/aloha_docs/2.0/operation/stationary.html

### Solo

| | |
|---|---|
| Dimensions | 1019D x 1066H x 1225W mm |
| Leader Arms | WidowX 250 S - Aloha Version |
| Follower Arms | ViperX 300 S - Aloha Version |
| Camera | 2x Intel RealSense D405 |
| Chassis | Modular |
| Computer | Coming Soon |
| USB Hubs | Yes 1X |

https://docs.trossenrobotics.com/aloha_docs/2.0/specifications.html#aloha-solo

https://docs.trossenrobotics.com/aloha_docs/2.0/operation/solo.html

## Trossen Robotics (Interbotix) X-Series Arms

https://docs.trossenrobotics.com/interbotix_xsarms_docs/specifications.html

ALOHA WidowX-250 6DOF

ALOHA ViperX-300 6DOF

## Waveshare RoArm

- 5DOF + pince Waveshare
- https://github.com/waveshareteam/roarm_ws
- https://www.waveshare.com/product/roarm-m3.htm?sku=30444

## ROBOTIS OMX

Follower:

- 5DOF + pince
- https://ai.robotis.com/omx/introduction_omx.html

Leader:

## ROBOTIS Open Manipulator-P

- 5DOF + pince
- Modbus-RTU
- https://emanual.robotis.com/docs/en/platform/openmanipulator_p/overview/

## ROBOTIS Open Manipulator-X

https://emanual.robotis.com/docs/en/platform/openmanipulator_x/specification/#specification

- 4 DOF Manipulateur + 1 DOF Pince
- 6x Dynamixel XM430-W350 https://emanual.robotis.com/docs/en/dxl/x/xm430-w350/
- Carte de contrôle Robotis OpenCR1.0
  https://emanual.robotis.com/docs/en/parts/controller/opencr10/

## SO-ARM100

https://github.com/TheRobotStudio/SO-ARM100

- 5 DOF Manipulateur + 1 DOF Pince
- 6 servos Feetech STS3215 https://www.feetechrc.com/en/2020-05-13_56655.html
- Waveshare Serial Bus Servo Driver Board
  https://www.waveshare.com/wiki/Bus_Servo_Adapter_(A)
- OU
- Feetech FE-URT-1 https://www.feetechrc.com/FE-URT1-C001.html

https://github.com/huggingface/lerobot/blob/main/examples/10_use_so100.md

https://medium.com/@sarohapranav/my-experiences-and-tips-for-creating-a-robotic-so100-arm-3df779a4aae7

https://github.com/JafarAbdi/ros2_so_arm100

## pince compatible SO-ARM

- Waveshare https://www.waveshare.com/gripper-a.htm?sku=30386

## Poppy Ergo JR

6DOF mais architecture semble optimisée pour faible le coût, qui n'a plus trop de sens avec le coût des servos Feetech. Une architecture 5DOF ou "typique industrielle" de type épaule poignet semble plus intéressante.

https://www.poppy-education.org/robots/poppy-ergo-jr/

# Cartes de contrôle

## OpenCR1.0

https://emanual.robotis.com/docs/en/parts/controller/opencr10/

- STM32F746ZGT6 / 32-bit ARM Cortex®-M7 with FPU (216MHz, 462DMIPS)
  Reference Manual, Datasheet
- Programmer : ARM Cortex 10pin JTAG/SWD connector
  USB Device Firmware Upgrade (DFU)
  Serial
- Digital I/O
  - 32 pins (L 14, R 18) *Arduino connectivity
  - 5Pin OLLO x 4
  - GPIO x 18 pins

    - PWM x 6
    - I2C x 1
    - SPI x 1
- Communication Ports
    - USB x 1 (Micro-B USB connector/USB 2.0/Host/Peripheral/OTG)
    - TTL x 3 (B3B-EH-A / DYNAMIXEL)
    - RS485 x 3 (B4B-EH-A / DYNAMIXEL)
    - UART x 2 (20010WS-04)
    - CAN x 1 (20010WS-04)

## Waveshare Serial Bus Servo Driver Board

https://www.waveshare.com/wiki/Bus_Servo_Adapter_(A)

- Supports connecting to a host or MCU
- up to 253 ST/SC series serial bus servos
- RS485
- UART pour contrôle depuis Arduino, ESP32, STM32 (RX-RX, TX-TX)
- USB pour contrôle via Raspberry, Jetson ou PC
- 9~12.6V voltage input (the input voltage and the servo voltage must be matched)

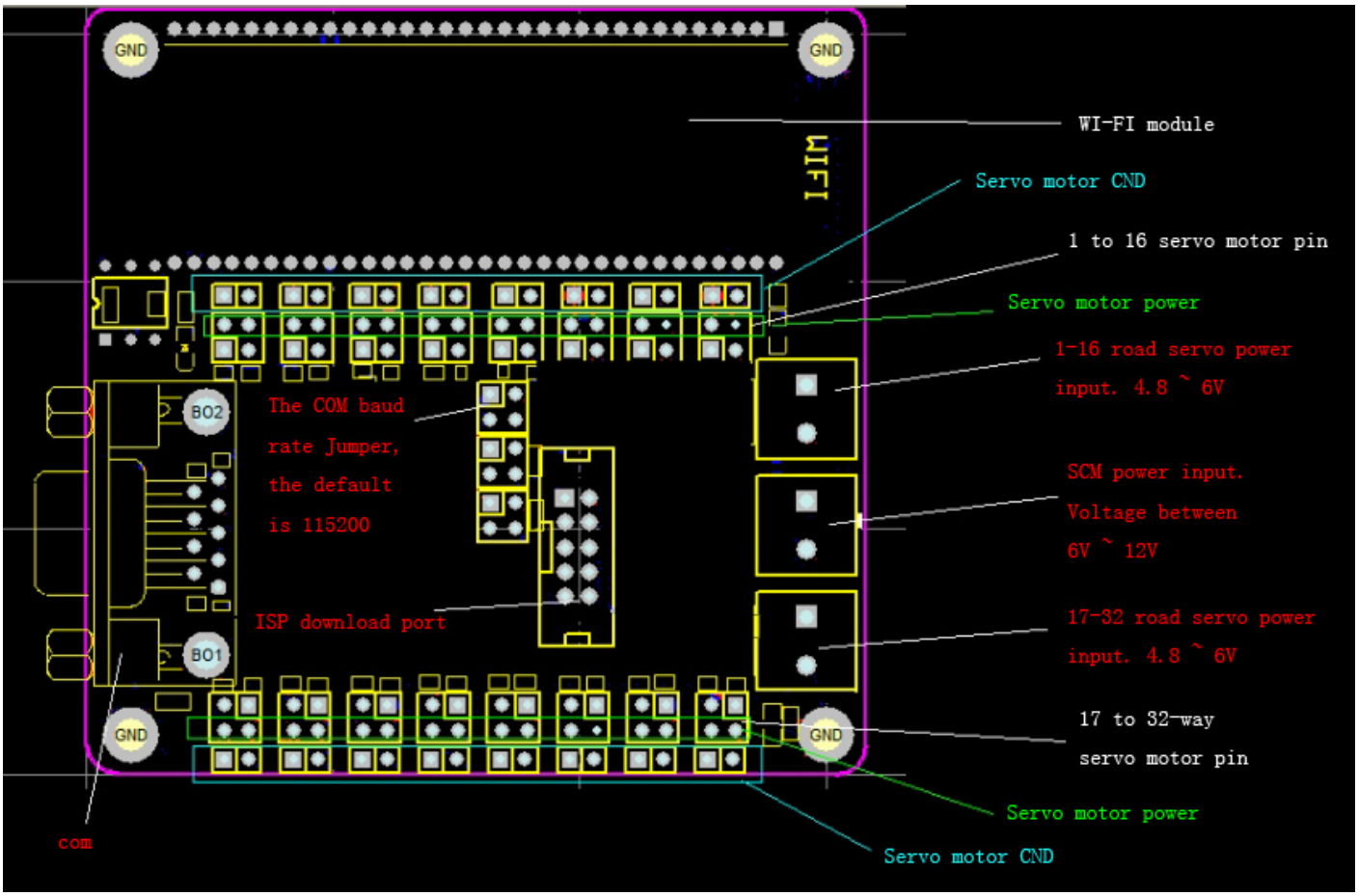
## Feetech FE-URT-1

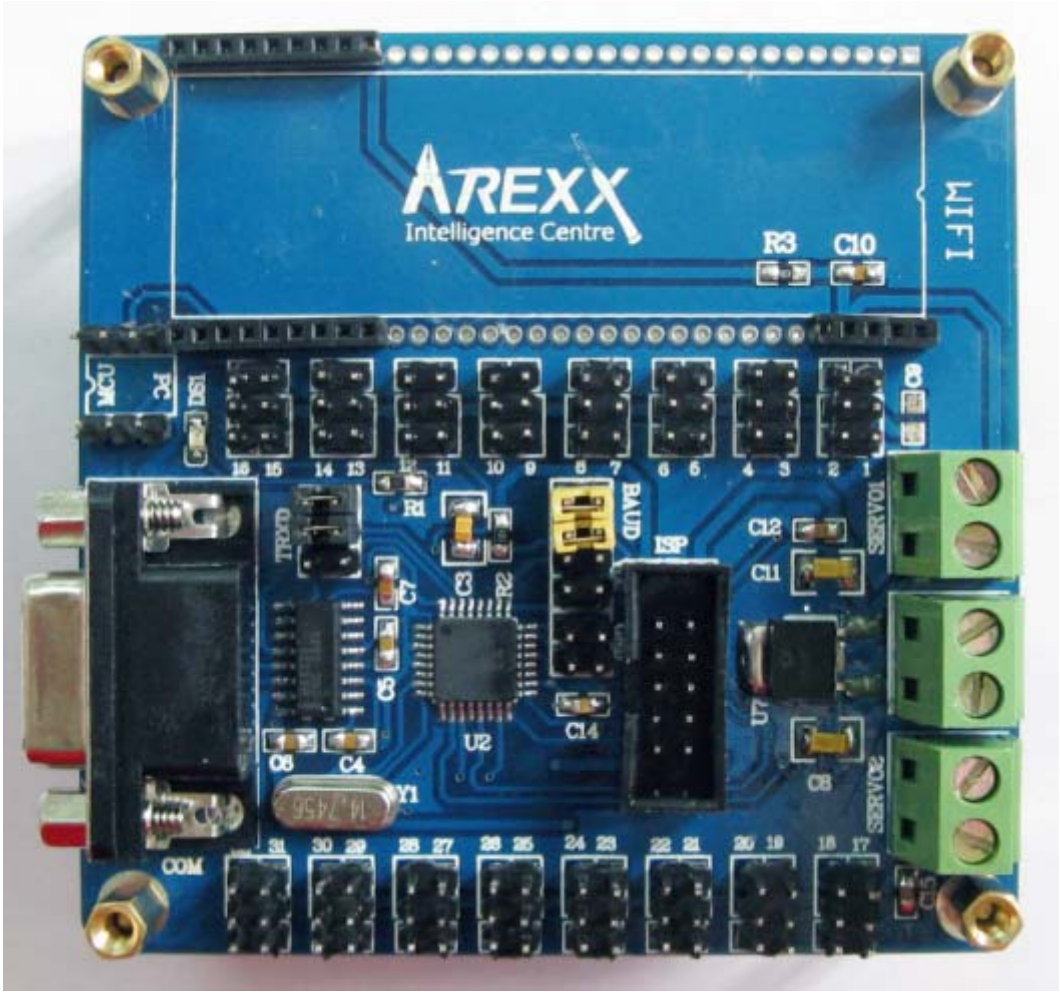
https://www.feetechrc.com/FE-URT1-C001.html

## AREXX Intelligence Centre

https://seafile.unistra.fr/d/693101e6046d4819a3af/

- atmega168 MCU
- RS232
- default baud rate is 115.2k
- Wifi wireless control reserve the ISP downloaded, you can download the MCU controller program using the STK500 ISP cable


GND
GND
WI-FI module
WIFI
Servo motor CND
1 to 16 servo motor pin
Servo motor power
1-16 road servo power
input. 4.8 ~ 6V
BO2
The COM baud
rate Jumper,
the default
is 115200
SCM power input.
Voltage between
6V ~ 12V
ISP download port
17-32 road servo power
input. 4.8 ~ 6V
BO1
GND
GND
17 to 32-way
servo motor pin
Servo motor power
com
Servo motor CND


AREXX
Intelligence Centre
R3
C10
WIFI
R1
BAUD
ISP
C12
C11
U2
C14
C8
C4
Y1
COM

- dual - Power Supply
  - 6 ~ 12 V SCM power
  - 4.8 ~ 6 V, 1.2A servo motor power [servo motor power supply Road 1-16 respectively, a 17-32 road supply port])

# Servomoteurs

## Dynamixel XM430-W350

https://emanual.robotis.com/docs/en/dxl/x/xm430-w350/

- 4.1 [N.m] (at 12.0 [V], 2.3 [A])
- 46 [rev/min] (at 12.0 [V])
- 10.0 ~ 14.8 [V]
- Operating Modes
  - Current Control Mode
  - Velocity Control Mode
  - Position Control Mode (0 ~ 360 [°])
  - Extended Position Control Mode (Multi-turn)
  - Current-based Position Control Mode
  - PWM Control Mode (Voltage Control Mode)
- baud rate 9,600 [bps] ~ 4.5 [Mbps]
- TTL Half Duplex Asynchronous Serial Communication with 8bit, 1stop, No Parity
- RS485 Asynchronous Serial Communication with 8bit, 1stop, No Parity

## Feetech STS3215

https://www.feetechrc.com/en/2020-05-13_56655.html

# SO-ARM100 - Robotique éducative

## Les bases d'un bras robot

https://docs.phospho.ai/learn/overview

## Pilotage bras robot avec Scratch

https://www.poppy-education.org/activites/initiation-ergo-jr-et-scratch/

## Pilotage du SO-ARM100 avec Phospho

https://docs.phospho.ai/learn/overview

## Simulation et pilotage du SO-ARM100 avec ROS2

> Attention avant d'utiliser le robot avec ROS2, il faut avoir calibré les servomoteurs, par ex. avec le script de calibration du projet LeRobot

https://github.com/JafarAbdi/ros2_so_arm100

### Jumeau numérique

Pilotage de la simulation ou du vrai robot

https://github.com/tessel-la/robo-boy

Adapter le tuto suivant au SO-ARM100 : https://innovation.iha.unistra.fr/books/robotique-open-source/page/programmer-un-robot-avec-moveit2-jumeau-numerique

## Contrôle des moteurs par un GUI de "jogging"

Joint Trajectory Controller

`ros2 run rqt_joint_trajectory_controller rqt_joint_trajectory_controller`

https://github.com/tessel-la/robo-boy

## Contrôle de l'outil sans collisions via le plugin MoveIt de RViz

Cartesian Trajectory

`ros2 launch so_arm100_moveit_config moveit_rviz.launch.py`

## Réalisation d'un programme en Python

https://moveit.picknik.ai/main/doc/examples/motion_planning_python_api/motion_planning_python_api_tutorial.html#single-pipeline-planning-pose-goal

```
# set plan start state to current state
panda_arm.set_start_state_to_current_state()

# set pose goal with PoseStamped message
pose_goal = PoseStamped()
pose_goal.header.frame_id = "panda_link0"
pose_goal.pose.orientation.w = 1.0
pose_goal.pose.position.x = 0.28
pose_goal.pose.position.y = -0.2
pose_goal.pose.position.z = 0.5
panda_arm.set_goal_state(pose_stamped_msg=pose_goal, pose_link="panda_link8")

# plan to goal
plan_and_execute(panda, panda_arm, logger)
```

En utilisant l'environnement de développement Jupyter Notebook
https://moveit.picknik.ai/main/doc/examples/jupyter_notebook_prototyping/jupyter_notebook_prototyping_tutorial.html

# Pilotage du bras robot par LeRobot (IA, VR, etc.)

Environnement Python sous Windows ou Linux

# Contrôle du bras par clavier ou manette

Avec LeRobot+Phospho https://docs.phospho.ai/basic-usage/teleop

ou avec ROS2+MoveIt2
https://moveit.picknik.ai/main/doc/examples/jupyter_notebook_prototyping/jupyter_notebook_prototyping_tutorial.html

# Contrôle du bras par Oculus Quest

Depuis Windows :

- Appli Oculus Phospho https://docs.phospho.ai/examples/teleop
- 222€ https://www.meta.com/en-gb/experiences/phospho-teleoperation/8873978782723478/

Depuis Ubuntu avec ROS2 et moveit_servo :

- https://github.com/ZorAttC/franka_vr
- https://moveit.picknik.ai/main/doc/examples/realtime_servo/realtime_servo_tutorial.html
- https://github.com/rail-berkeley/oculus_reader
  - Enable Oculus Quest development mode
  - Always allow USB debugging from this computer
  - Connexion USB (ADB) ou wifi

Autre : https://github.com/lts0429/teleoperation

# Contrôle du bras via un modèle d'IA

https://docs.phospho.ai/basic-usage/inference

- Créer un compte huggingface.ia
- Sign In dans phosphobot
- Dans les paramètres, ajouter la clé d'API huggingface
- Par défaut l'inférence du modèle d'IA qui permet de piloter le robot depuis l'image des caméras tournera sur un GPU sur les serveurs de huggingface ou phospho
- On peut faire tourner l'inférence du modèle d'IA sur le PC local s'il a une bonne carte graphique NVidia

- Suivre ces instructions : https://github.com/phospho-app/phosphobot/tree/main/inference#setup-a-server
- Démarrer le serveur d'inférence uv
- `uv run ACT/server.py --model_id=<CHEMIN_VERS_LE_MODELE_LOCAL>`
- Appeler le serveur d'inférence depuis un script python : https://docs.phospho.ai/basic-usage/inference#2-call-your-inference-server-from-a-python-script

# Pilotage ST3215 depuis un ESP32 (embarqué, micro-ROS)

## Sans utiliser la carte de contrôle Feetech/Waveshare

(5€) : https://github.com/sepastian/ESP32_ST3215

## Pilotage bluetooth depuis un smartphone

Utiliser l'ESP32 pour faire l'interface bluetooth vers une télécommande smartphone ? Avec ou sans carte de contrôle officielle (cf. ci-dessus) ?

## Micro-ROS avec bras robot

- approche pour grasping référencée capteur via un TOF-sensor

https://micro.ros.org/docs/tutorials/demos/openmanipulator_demo/

- Télécommander une pose relative de la pince via un capteur https://micro.ros.org/docs/tutorials/demos/moveit2_demo/

> 6-DoF Inertial Measurement Unit (LSM6DSL), composed of an accelerometer and a gyroscope, and a 3-DoF magnetometer (LIS3MDL). The fusion of the measurements fetched by these sensors outputs the pose, or relative orientation of the board with respect to a fixed reference frame.

# ROS2 avec SO-ARM101

## ROS2 et MoveIt2

Installer les paquets ROS2 du SO-ARM100 :

- Cloner le paquet dans un workspace ROS2 https://github.com/JafarAbdi/ros2_so_arm100
- Cloner le submodule https://github.com/TheRobotStudio/SO-ARM100 dans `so_arm100_description/SO-ARM100` (https://www.freecodecamp.org/news/how-to-use-git-submodules/)
- Ou simplement :

```
mkdir -p ~/ws_so_arm100/src
cd ~/ws_so_arm100/src
git clone --recurse-submodules https://github.com/JafarAbdi/ros2_so_arm100
cd ~/ws_so_arm100
sudo rosdep init
rosdep update && rosdep install --ignore-src --from-paths src -y
colcon build --symlink-install # dans une VM ajouter --parallel-workers 1
source install/setup.bash
ros2 launch so_arm100_moveit_config demo.launch.py hardware_type:=mock_components #
hardware_type:=real for running with hardware
```

Tester la démo en simulation :

- Lancer un des scripts : https://github.com/JafarAbdi/ros2_so_arm100?tab=readme-ov-file#usage

# IA robotique - Architectures pour l'apprentissage profond

Le contrôle d'un robot pour une application donnée au moyen d'un modèle de réseaux de neurones nécessite de fournir une quantité importante de données d'apprentissage. Ces données doivent permettre de comprendre comment résoudre la tâche, par exemple pour une tâche de saisie et dépose d'un objet (Pick and Place) elles doivent donc comporter :

- Comment bouge le robot et chacun de ses moteurs
- Comment bouge la pince
- Où sont placés les objets au début "problème"
- Où sont placé les objets à la fin "solution"

L'approche la plus répandue pour générer ces données d'apprentissage est la démonstration : le robot est "téléguidé" par un opérateur. On enregistre la trajectoire des servomoteurs ainsi qu'une ou plusieurs vidéos filmant les objets et le robot.

## Bancs matériel de Machine Learning

### Koch

### Robotis OMX

https://ai.robotis.com/omx/introduction_omx.html

### LeRobot SO-ARM10X

# Aloha Solo

Base : $9,899.95

- WidowX Leader Arm ($4,949.95)
- ViperX Follower Arm ($7,149.95)
- 2x Intel RealSense D405 Cameras
- Portable Touchscreen Monitor
- Tripod, Cables, Accessories

https://www.youtube.com/watch?v=hFqZJZ666Cw

https://www.trossenrobotics.com/aloha-solo

# Aloha Stationary

Without Laptop : $30,799.98

# Environnement logiciel de collecte de données

## Hugging Face LeRobot - Python

Visualisation des DataSet : https://huggingface.co/spaces/lerobot/visualize_dataset

On note qu'il y a beaucoup de tests avec le SO-ARM10X. Mais pas d'homogénéité dans la collecte de données : les caméras sont placées la plupart du temps devant le robot, parfois sur la pince, et quasiment jamais avec le setup recommandé.

Rechercher des DataSet qui ont servi à entrainer des modèles disponibles publiés :

- https://huggingface.co/models?pipeline_tag=robotics&sort=trending
- Dans le panneau de droite > Sélectionner le filtre Reinforcement Learning > Robotics
- Beaucoup de DataSets sont uploadés mais peu de modèles entrainés le sont
- Par exemple on recherche les travaux de Rémi Cadene
  - Modèles : il y en a 4

    https://huggingface.co/models?pipeline_tag=robotics&sort=trending&search=caden

e
    - DataSets : Il y en a beaucoup https://huggingface.co/datasets?task_categories=task_categories:robotics&sort=trending&search=cadene
- Les Modèles entrainés avec le SO-ARM10 concernent la manipulation de Lego, par exemple le plus récent :
    - Modèle : https://huggingface.co/cadene/act_so100_5_lego_test_080000
    - DataSet pour le training : https://huggingface.co/datasets/cadene/so100_5_lego_test
    - DataSet pour l'évaluation du modèle : https://huggingface.co/datasets/cadene/eval_act_so100_5_lego_test_080000

Setup similaires à celui du SO-ARM10X simple :

- Robot Koch : /lerobot/koch_pick_place_1_lego

Télécharger un DataSet : https://github.com/huggingface/lerobot/blob/a1daeaf0c4ae345df9b2f5b862f091ce158e4446/examples/1_load_lerobot_dataset.py

Rejouer les épisodes d'un DataSet : https://github.com/huggingface/lerobot/blob/a1daeaf0c4ae345df9b2f5b862f091ce158e4446/examples/12_use_so101.md#replay-an-episode

Evaluer un modèle pré-entrainé : https://github.com/huggingface/lerobot/blob/a1daeaf0c4ae345df9b2f5b862f091ce158e4446/examples/2_evaluate_pretrained_policy.py

# Trossen Robotics Interbotix - ROS, Python,

# Machine Learning LeRobot avec SO-ARM101

## Installation et prérequis

- Une carte graphique NVidia et une installation de Cuda ?
- Comparaison cartes graphiques (un peu dépassé car ne prend pas en compte les dernières génération type 5090) https://www.geeksforgeeks.org/machine-learning/choosing-the-right-gpu-for-your-machine-learning/
- Puissance de calcul dispo à l'Innov'Lab TPS https://www.innovlab-tps.net/calcul-et-stockage.html
  - Pour l'**entraînement** des modèles : Serveur de calcul Apollo 6500 doté de **4 GPU HGX A100, 80Go**
  - Pour l'**inférence** des modèles : Terminaux de calcul dotés GPU Nvidia **RTX 4090, 24Go**

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

- Config minimum : L'exécution semble résulter en un mouvement saccadé du robot sur une Quadro P620 (2 GB GDDR5).
- Config recommandée : 4-8 GB GDDR, testé avec RTX 2080 Super de 2019 (8 GB GDDR6)
- Il faut au moins 16 GB de RAM (CPU) sinon elle sature pendant l'enregistrement du dataset d'évaluation.

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

- L'entrainement avec 100 épisodes et 100 000 steps a mis 12H sur une RTX 2080 Super de 2019
  - mais `batch_size=8` alors qu'il faut un minimum de 16, et dans l'idéal 64. Cela résulte sans doute en des mouvements saccadés et une mauvaise généralisation du modèle (variation de la position de la pile).
  - Il n'aboutit pas au bout de plus de 24H sur une Quadro P620 (via WSL2 et Docker)
- Config minimum pour ACT : `batch_size=16`
  - Tesla T4 (gratuit 5H / mois sur Google Colab) on a 15G de RAM qui permet donc un `batch_size=15`
  - https://github.com/huggingface/lerobot/issues/2213
- Config recommandée pour ACT : `batch_size=64`

- avec A100 (12€ les 100 compute units, 70 = ~5H sur Google Colab) on a 40G de RAM ou 80G de RAM (extra RAM)
- https://github.com/huggingface/lerobot/blob/main/docs/source/notebooks.mdx#training-act
- https://colab.research.google.com/github/antonilo/real_world_robot_learning_sp25/blob/main/_tutorials/lerobot_tutorial/lerobot_tutorial.ipynb#scrollTo=ff7eafe4
- avec un `batch_size=64` la RAM consommée était d'environ 50G, l'entraînement a mis 7H environ pour environ 80 compute units, soit environ 10€
- https://huggingface.co/gautz/act_so101_pick_red_18650_drop_black_box_test2_100ep_64batch_60ksteps
- https://wandb.ai/hentz-robotics/lerobot/runs/niiti0v3?nw=nwusergautz

# 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]"
```

# Débuguer l'installation si les scripts basculent sur le `cpu`

S'assurer que Pytorch a été installé, cela installe Cuda :

`pip3 install --pre torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu128`

Récupérer les infos système :

- `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.

# Créer un compte HuggingFace et se login via un token

- Se créer un compte sur https://huggingface.co/settings/tokens
- Aller dans le menu > Access Tokens
- Récupérer un `TOKEN` avec des droits en écriture
- Ouvrir un Anaconda PowerShell lerobot
  `(lerobot) PS C:\Users\gauthier.hentz\github\lerobot>`
- Login avec le Token
  `huggingface-cli.exe login --token TOKEN`

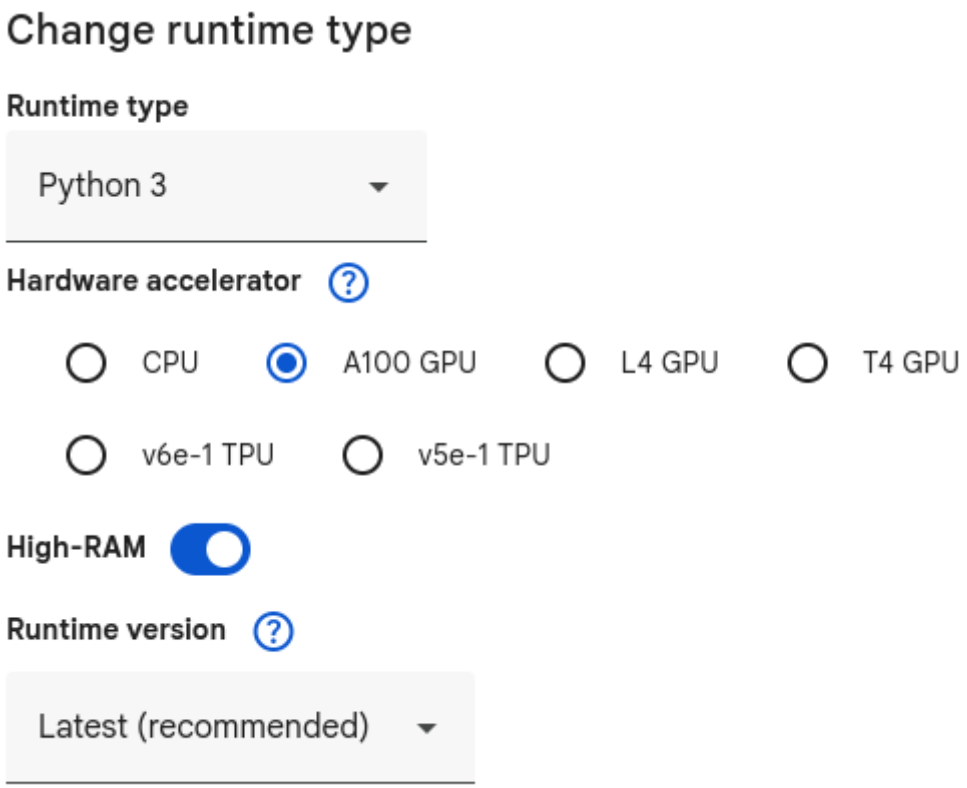
Vous pouvez maintenant uploader et downloaded des DataSets et Modèles vers le Hub HuggingFace pour collaborer avec des experts du Machine Learning.

# Google Colab

Pour avoir accès à un GPU avec suffisamment de mémoire (16G recommandé et 64G dans l'idéal) on peut utiliser Google Colab

- Avec Tesla T4 (gratuit 5H / mois) on a 15G de RAM qui permet donc un `batch_size=15`
- Avec A100 (12€ les 70 crédits) on a 80G de RAM qui permet donc un `batch_size=64`

- Ouvrir le Notebook :
  https://colab.research.google.com/github/huggingface/notebooks/blob/main/lerobot/training-act.ipynb#scrollTo=NQUk3Y0WwYZ4
- Voir aussi
  https://colab.research.google.com/github/antonilo/real_world_robot_learning_sp25/blob/main/_tutorials/lerobot_tutorial/lerobot_tutorial.ipynb#scrollTo=ff7eafe4
- Change runtime type




- Install conda
- Install LeRobot
- Login avec un jeton d'API Weights & Biases
- Login avec un jeton d'API Hugging Face Hub
- Modifier la commande d'entrainement et l'exécuter :

```
!cd lerobot && python src/lerobot/scripts/lerobot_train.py \
  --dataset.repo_id=gautz/so101_pick_red_18650_drop_black_box_test2_100ep \
  --policy.type=act \
  --output_dir=outputs/train/act_so101_pick_red_18650_drop_black_box_test2_100ep_64batch \
  --job_name=act_so101_pick_red_18650_drop_black_box_test2_100ep_64batch \
  --policy.device=cuda \
  --policy.push_to_hub=True \
  --policy.repo_id=gautz/act_so101_pick_red_18650_drop_black_box_test2_100ep_64batch \
  --batch_size=64 \
  --wandb.enable=true
```

- Surveiller l'éxecution depuis le runtime ou WanDB
- Ouvrir un Terminal et lancer la commande pour uploader le dernier Checkpoint du modèle :

```
huggingface-cli upload ${HF_USER}/act_so101_pick_red_18650_drop_black_box_test2_100ep_64batch \
```

```
/content/lerobot/outputs/train/act_so101_pick_red_18650_drop_black_box_test2_100ep_64batch/checkpoi
```

`nts/last/pretrained_model` https://api.wandb.ai/links/hentz-r

Astuces et Erreurs :

- Si vous utilisez Colab avec un compte gratuit, il faut utiliser l'extension Colab Auto Reconnect pour ne pas être déconnecté au boût de 1H. Il est probable que le runtime soit déconnecté par manque de crédit gratuit avant d'arriver à un checkpoint. Faire des CheckPoint plus souvent, ou passer à un compte payant.
- Si vous utilisez un compte payant, vous ne serez pas déconnecté mais il faut surveiller pour déconnecter le Runtime à la fin de l'entraînement (ou d'un Checkpoint)
- https://github.com/huggingface/lerobot/issues/1532

```
cd /content/
cd lerobot/
python
            from huggingface_hub import HfApi
            hub_api = HfApi()
            hub_api.create_tag("gautz/so101_pick_red_18650_drop_black_box_test2_100ep", tag="v2.1", repo_type="dataset"
cd src/
cd lerobot
python -m lerobot.datasets.v30.convert_dataset_v21_to_v30 --repo-id=gautz/so101_pick_red_18650_drop_black_box_test2_100ep
```

- https://stackoverflow.com/questions/78448832/colab-how-to-increase-the-num-workers-in-dataloader

Voir le rapport : https://api.wandb.ai/links/hentz-robotics/wdsr4k2r

# Utilisation de LeRobot avec le bash Linux

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

`lerobot-find-port`

A chaque connexion du robot : `sudo chmod 666 /dev/ttyACM1`

`lerobot-find-cameras opencv`

`lerobot-calibrate --robot.type=so101_follower --robot.port=/dev/ttyACM0 --robot.id=follower_arm_fan1`

`lerobot-calibrate --teleop.type=so101_leader --teleop.port=/dev/ttyACM1 --teleop.id=leader_arm_fan1`

`lerobot-teleoperate --robot.type=so101_follower --robot.port=/dev/ttyACM0 --robot.id=follower_arm_fan1 --teleop.type=so101_leader --teleop.port=/dev/ttyACM1 --teleop.id=leader_arm_fan1`

`lerobot-dataset-viz --repo-id gautz/so101_pick_red_18650_drop_black_box_test2_100ep --episode-index 0`

`lerobot-record --robot.type=so101_follower --robot.port=/dev/ttyACM0 --robot.id=follower_arm_fan1 --robot.cameras="{ top: {type: opencv, index_or_path: 4, width: 800, height: 600, fps: 20},follower: {type: opencv, index_or_path: 2, width: 800, height: 600, fps: 20}}" --display_data=true --dataset.push_to_hub=False --dataset.repo_id=gautz/eval_act_so101_pick_red_18650_drop_black_box_test2_100ep_64batch_60ksteps --dataset.num_episodes=10 --dataset.single_task="Pick red 18650 battery place black box" --policy.path=gautz/act_so101_pick_red_18650_drop_black_box_test2_100ep_64batch_60ksteps`

`--resume=true`

`lerobot-train --dataset.repo_id=gautz/so101_pick_red_18650_drop_black_box_test2_100ep --policy.type=act --output_dir=outputs/train/act_so101_pick_red_18650_drop_black_box_test2_100ep_m620 --job_name=act_so101_pick_red_18650_drop_black_box_test2_100ep_m620 --policy.device=cuda --wandb.enable=false --policy.repo_id=gautz/act_so101_pick_red_18650_drop_black_box_test2_100ep_m620 --batch_size=2`

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
```

Entrainer 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
```

Entrainer en local avec le GPU NVidia

```
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
```

# Utilisation de LeRobot avec Anaconda Powershell Prompt

## Calibration des robots

`python -m lerobot.setup_motors --robot.type=so101_follower --robot.port=COM13`
`python -m lerobot.setup_motors --robot.type=so101_leader --robot.port=COM14`

# Machine Learning

Collecte de données :

Uploader les données vers le Hub HuggingFace :

https://huggingface.co/docs/lerobot/en/il_robots#dataset-upload

```
huggingface-cli.exe upload gautz/so101_pick_red_18650_drop_black_box_test2_100ep
..\.cache\huggingface\lerobot\gautz\18650-test2-100ep\ --repo-type dataset
```

Entrainer un modèle sur un Dataset (sur `IHA-QLIOVR-1`, compte admin) :

```
cd ..\gauthier.hentz\lerobot\
conda activate lerobot
cd .\lerobot\
conda create -y -n lerobot python=3.10
conda activate lerobot
conda install ffmpeg -c conda-forge
ffmpeg -encoders
conda install ffmpeg=7.1.1 -c conda-forge
pip install av poetry-core
pip3 install --pre torch torchvision torchaudio --index-url
https://download.pytorch.org/whl/cu128
pip install -e .
python lerobot/scripts/train.py --dataset.repo_id=gautz/18650-test2-100ep --policy.type=act --
output_dir=outputs/train/act_so101_18650-test2-100ep --job_name=act_so101_18650-test2-100ep --
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
```

# Uploader un modèle vers HuggingFace

https://github.com/huggingface/lerobot/?tab=readme-ov-file#add-a-pretrained-policy

- Rechercher le dossier du Checkpoint voulu, par ex. `C:\Users\User\github\lerobot\outputs\train\act_so101_18650-test2-100ep\checkpoints\100000`
- Définir le User HuggingFace (qui doit être Login via un Token) `gautz` et le Repo `act_so101_pick_red_18650_drop_black_box_test2_100ep_100000` , puis uploader le Checkpoint : `huggingface-cli.exe upload gautz/act_so101_pick_red_18650_drop_black_box_test2_100ep_100000 .\outputs\train\act_so101_18650-test2-100ep\checkpoints\100000`

# Historique du Terminal complet

```
wget "https://repo.anaconda.com/miniconda/Miniconda3-latest-Windows-x86_64.exe" -outfile
".\Downloads\Miniconda3-latest-Windows-x86_64.exe"
cd .\github\lerobot\
conda create -y -n lerobot python=3.10
conda activate lerobot
conda install ffmpeg -c conda-forge
conda install ffmpeg=7.1.1 -c conda-forge
wsl --shutdown
pip install -e .
pip install -e ".[feetech]" # or "[dynamixel]" for example
python lerobot/find_port.py
python lerobot/find_cameras.py opencv
python -m lerobot.setup_motors --robot.type=so101_follower --robot.port=COM13`
python -m lerobot.record  --robot.type=so101_follower --robot.port=COM13 --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
python -m lerobot.calibrate     --teleop.type=so101_leader  --teleop.port=COM14 --
teleop.id=leader_arm_fan1`
python -m lerobot.calibrate     --robot.type=so101_follower  --robot.port=COM13 --
robot.id=follower_arm_fan1
python -m lerobot.teleoperate   --robot.type=so101_follower   --robot.port=COM13   --
robot.id=follower_arm_fan1   --robot.cameras="{ top: {type: opencv, index_or_path: 1, width:
640, height: 480, fps: 30},follower: {type: opencv, index_or_path: 2, width: 640, height: 480,
fps: 30}}"   --teleop.type=so101_leader   --teleop.port=COM14   --teleop.id=leader_arm_fan1
--display_data=true
```

```
python lerobot/find_cameras.py opencv
python -m lerobot.teleoperate   --robot.type=so101_follower   --robot.port=COM13   --robot.id=follower_arm_fan1   --robot.cameras="{ top: {type: opencv, index_or_path: 1, width: 800, height: 600, fps: 30},follower: {type: opencv, index_or_path: 2, width: 800, height: 600, fps: 30}}"   --teleop.type=so101_leader   --teleop.port=COM14   --teleop.id=leader_arm_fan1 --display_data=true
pip install -e .
python lerobot/find_cameras.py opencv
python -m lerobot.teleoperate   --robot.type=so101_follower   --robot.port=COM13   --robot.id=follower_arm_fan1   --robot.cameras="{ top: {type: opencv, index_or_path: 1, width: 800, height: 600, fps: 30},follower: {type: opencv, index_or_path: 0, width: 800, height: 600, fps: 30}}"   --teleop.type=so101_leader   --teleop.port=COM14   --teleop.id=leader_arm_fan1 --display_data=true
python -m lerobot.record  --robot.type=so101_follower --robot.port=COM13 --robot.cameras="{ top: {type: opencv, index_or_path: 1, width: 800, height: 600, fps: 30},follower: {type: opencv, index_or_path: 0, 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
python lerobot/scripts/display_sys_info.py
python -m torch.utils.collect_env
python -c "import torch; print(torch.cuda.is_available())" && nvcc -V
python -c "import torch; print(torch.cuda.is_available())"
pip3 install --pre torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu128
python -m lerobot.record  --robot.type=so101_follower --robot.port=COM13 --robot.cameras="{ top: {type: opencv, index_or_path: 1, width: 800, height: 600, fps: 30},follower: {type: opencv, index_or_path: 0, 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 --wandb.enable=false
python -m lerobot.record  --robot.type=so101_follower --robot.port=COM13 --robot.cameras="{ top: {type: opencv, index_or_path: 1, width: 800, height: 600, fps: 30},follower: {type: opencv, index_or_path: 0, 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     --dataset.episode_time_s=10  --dataset.reset_time_s=10    --
dataset.num_episodes=10
exit
cd .\github\lerobot\
conda activate lerobot
python -m lerobot.record  --robot.type=so101_follower --robot.port=COM13 --robot.cameras="{
top: {type: opencv, index_or_path: 1, width: 800, height: 600, fps: 30},follower: {type:
opencv, index_or_path: 0, 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     --dataset.episode_time_s=15  --dataset.reset_time_s=10    --
dataset.num_episodes=10
python -m lerobot.teleoperate   --robot.type=so101_follower   --robot.port=COM13   --
robot.id=follower_arm_fan1   --robot.cameras="{ top: {type: opencv, index_or_path: 1, width:
800, height: 600, fps: 30},follower: {type: opencv, index_or_path: 0, width: 800, height: 600,
fps: 30}}"   --teleop.type=so101_leader   --teleop.port=COM14   --teleop.id=leader_arm_fan1
--display_data=true
exit
cd .\github\lerobot\
conda activate lerobot
python -m lerobot.teleoperate   --robot.type=so101_follower   --robot.port=COM13   --
robot.id=follower_arm_fan1   --robot.cameras="{ top: {type: opencv, index_or_path: 1, width:
800, height: 600, fps: 30},follower: {type: opencv, index_or_path: 0, width: 800, height: 600,
fps: 30}}"   --teleop.type=so101_leader   --teleop.port=COM14   --teleop.id=leader_arm_fan1
--display_data=true
python lerobot/find_port.py
python -m lerobot.teleoperate   --robot.type=so101_follower   --robot.port=COM15   --
robot.id=follower_arm_fan1   --robot.cameras="{ top: {type: opencv, index_or_path: 1, width:
800, height: 600, fps: 30},follower: {type: opencv, index_or_path: 0, width: 800, height: 600,
fps: 30}}"   --teleop.type=so101_leader   --teleop.port=COM14   --teleop.id=leader_arm_fan1
--display_data=true
python lerobot/find_cameras.py opencv
python -m lerobot.teleoperate   --robot.type=so101_follower   --robot.port=COM15   --
robot.id=follower_arm_fan1   --robot.cameras="{ top: {type: opencv, index_or_path: 0, width:
800, height: 600, fps: 30},follower: {type: opencv, index_or_path: 2, width: 800, height: 600,
fps: 30}}"   --teleop.type=so101_leader   --teleop.port=COM14   --teleop.id=leader_arm_fan1
--display_data=true
python -m lerobot.record  --robot.type=so101_follower --robot.port=COM15 --robot.cameras="{
```

```
top: {type: opencv, index_or_path: 1, width: 800, height: 600, fps: 30},follower: {type: opencv, index_or_path: 0, 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     --dataset.episode_time_s=15  --dataset.reset_time_s=10    --dataset.num_episodes=10
python -m lerobot.record  --robot.type=so101_follower --robot.port=COM15 --robot.cameras="{ top: {type: opencv, index_or_path: 0, width: 800, height: 600, fps: 30},follower: {type: opencv, index_or_path: 2, 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     --dataset.episode_time_s=15  --dataset.reset_time_s=10    --dataset.num_episodes=10
python -m lerobot.teleoperate   --robot.type=so101_follower   --robot.port=COM15   --robot.id=follower_arm_fan1   --robot.cameras="{ top: {type: opencv, index_or_path: 0, width: 800, height: 600, fps: 30},follower: {type: opencv, index_or_path: 2, width: 800, height: 600, fps: 30}}"   --teleop.type=so101_leader   --teleop.port=COM14   --teleop.id=leader_arm_fan1 --display_data=true
cd .\github\lerobot\
cd ..
cd .\.cache\
cd .\huggingface\ls
cd .\huggingface\lerobot\calibration\cd ..
cd .\huggingface\lerobot\
cd .\gautz\
cd ..
cd .\github\lerobot\
cd .\outputs\train\
cd .\act_so101_18650-test2-100ep\
cd .\checkpoints\
```

# Migration dataset  v2.1 to v3

https://github.com/huggingface/lerobot/blob/main/docs/source/lerobot-dataset-v3.mdx#migrate-v21--v30

# Sources

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

https://huggingface.co/spaces/lerobot/visualize_dataset?path=%2Fsebastiandavidlee%2Fbimanual-so101-handover-plushie%2Fepisode_40

https://huggingface.co/blog/sherryxychen/train-act-on-so-101

https://github.com/sherrychen1120/so101_bench

https://www.linkedin.com/posts/sherryxychen_robotics-machinelearning-ai-ugcPost-7378831270207954944-WsI5?utm_source=share&utm_medium=member_desktop&rcm=ACoAAA1icaQBkYXRoM1Oslzrh8psuL0MmHpaT_4

# Pilotage des servomoteurs : TTL, RS232, RS485

## Modes de contrôle des servomoteurs

Regarder la classification des constructeurs permet de se rendre compte des différentes manières de piloter un servomoteur :

- Feetech https://www.feetechrc.com/
- Robotis :
  - https://www.robotis.fr/index.php?id_category=7&controller=category
  - https://emanual.robotis.com/docs/en/dxl/
  - https://www.dynamixel.com/
  - https://www.dynamixel.com/whatisdxl.php

Cela va donc du contrôle PWM jusqu'aux bus et protocoles industriels :

- Servos de modélisme asservis en position "servo 180°" ou en vitesse "servo 360°" via signal PWM
  - Feetech "PWM series servo"

    https://www.feetechrc.com/pwm%20series%20servo.html
  - https://arduino.blaisepascal.fr/conversion-numeriqueanalogique-pwm/
  - https://arduino.blaisepascal.fr/communication-2/
  - https://arduino.blaisepascal.fr/premiers-pas/faire-tourner-les-servos-2/
  - https://arduino.blaisepascal.fr/servo-suiveur/
  - https://arduino.blaisepascal.fr/les-servomoteurs/
  - https://arduino.blaisepascal.fr/controle-dun-servomoteur/
- Servos pédagogiques Dynamixel "série X" ou Feetech "Smart Serial Bus Servo"
  - TTL, ex. Feetech STS3235
  - RS485, ex. Feetech SMS..
- Servos professionnels Dynamixel "série P" ou Feetech "Modbus RTU Series Servo", par ex.
  - Modbus RTU https://celka.fr/ocw/plc-control/modbus/intro-modbus/intro/
  - Modbus TCP https://celka.fr/ocw/plc-control/modbus/modbus-tcp/modbus-tcp/

# Introduction au contrôle PLC

https://celka.fr/ocw/plc-control/modbus/intro-modbus/intro/

# Protocoles de communication

Dynamixel :

- Dynamixel Protocol 2.0 https://emanual.robotis.com/docs/en/dxl/protocol2/
- Modbus RTU pour les Dynamixel Pro (PH, RH, PM) https://emanual.robotis.com/docs/en/dxl/p/ph42-020-s300-r/#protocol-type13

Feetech :

- Modbus RTU pour les modèles : https://www.feetechrc.com/modbus-rtu%20series%20servo.html
  - Exemple servo 24V 24kg https://www.feetechrc.com/24v-24kgcm-modbus-rtu%E8%88%B5%E6%9C%BA.html

https://esp32io.com/tutorials/esp32-rs485

# Transmission TTL et protocole RS485

## Transmission série

Avantages

- Câble plus fin, plus souple, moins coûteux.
- Connecteur simplifié, meilleur marché, plus vite monté.
- Plus de problème de synchronisation de signaux
  - On ne transmet qu’un seul signal. Seules les horloges doivent être de fréquence très voisine, ce qui n’est pas difficile en électronique.
- Isolation diaphonique.
  - Plus de risque d’interférence entre signaux, il n’y a qu’un seul signal.
- Utilisable sur des longueurs nettement plus importantes (km).

Inconvénients

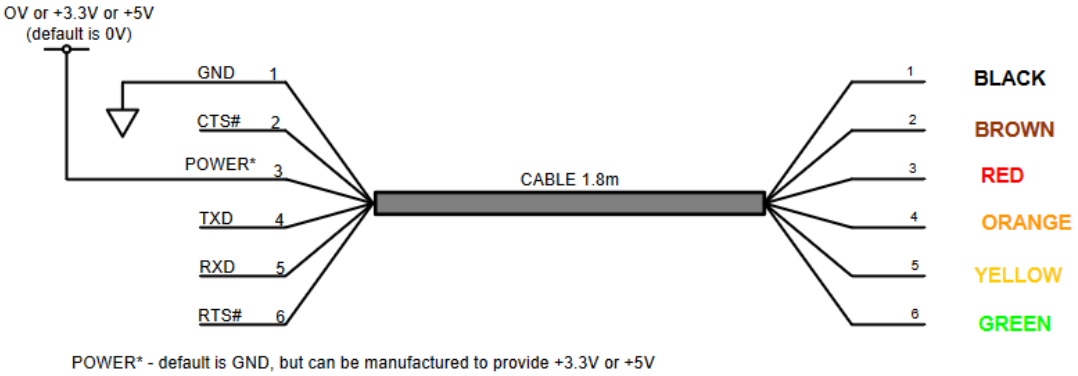
- Débit
  - A une même fréquence, on transporte un seul bit à la fois.
- Electronique plus compliquée du côté émetteur et encore plus compliquée côté récepteur (synchronisation d’horloge).
  - UART : Universal Asynchronous Receiver Transmitter.
  - L’UART peut être désynchronisé, l’information reçue est alors invalide.

## Transmission série synchrone

## Transmission série asynchrone

Exemple : port série RS232 du PC

Exemple d’un adapteur FTDI USB-RS232

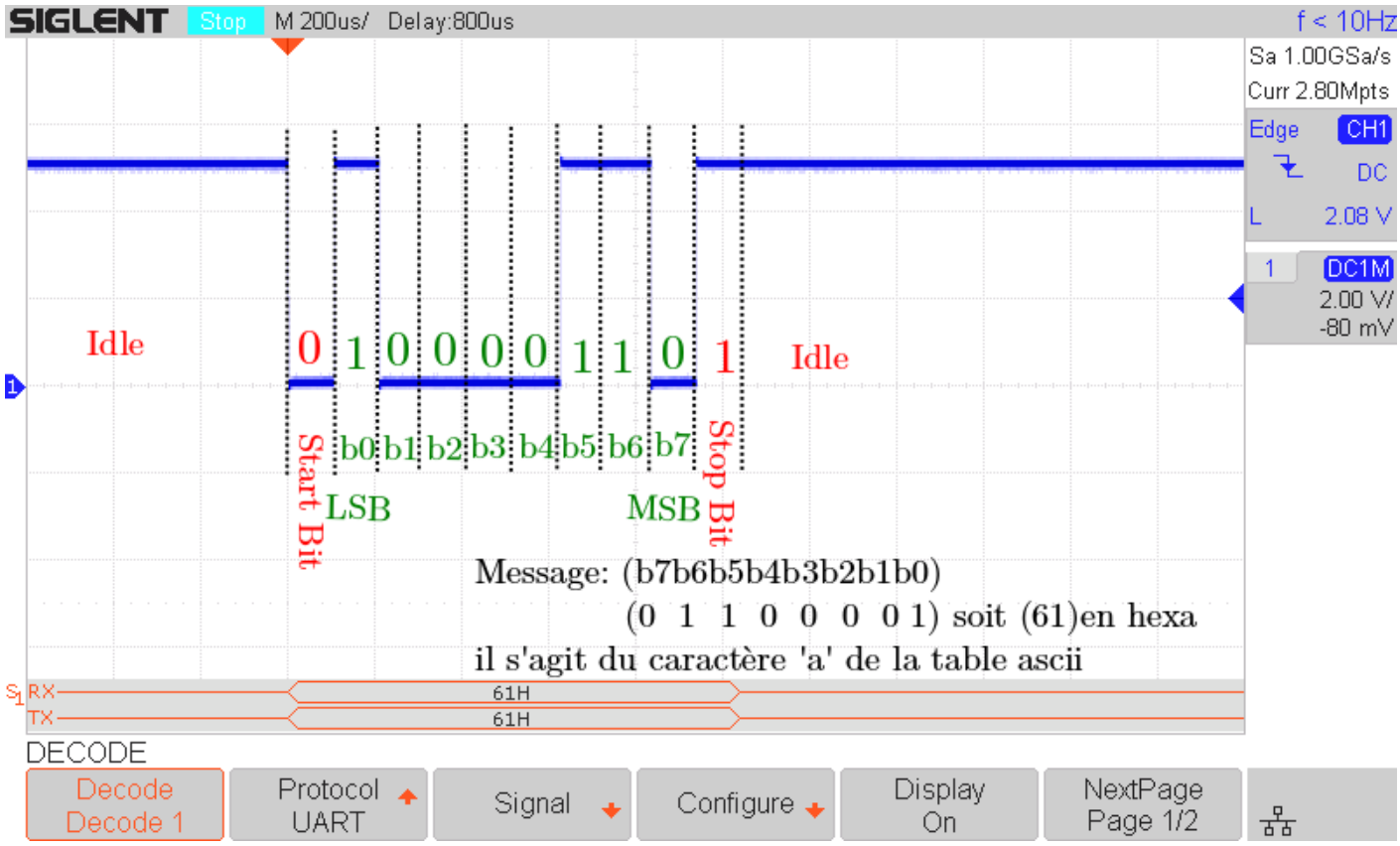



# Transmission série asynchrone TTL

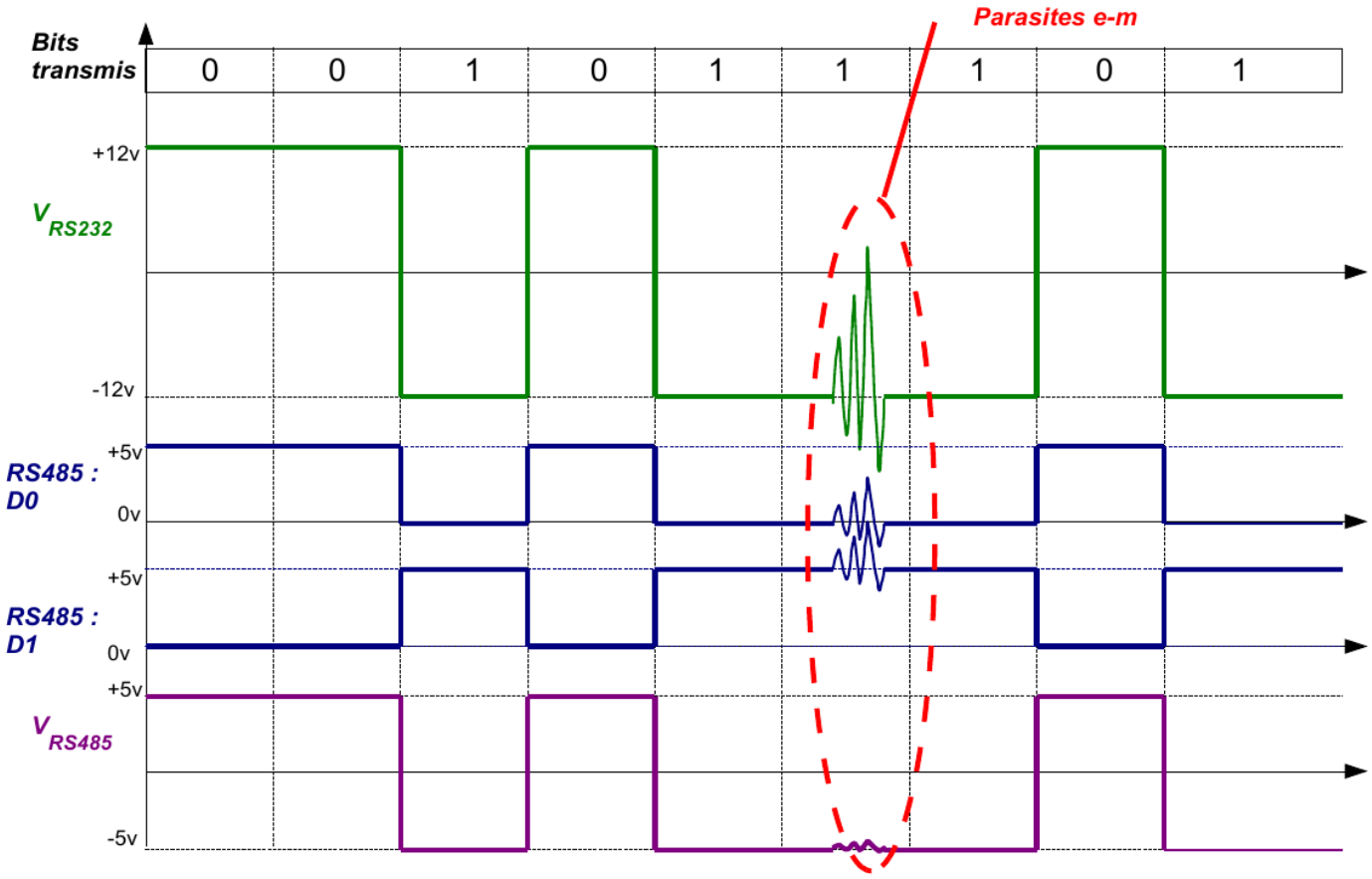
Exemple de trame série (TTL)

‘1’ logique = +5V

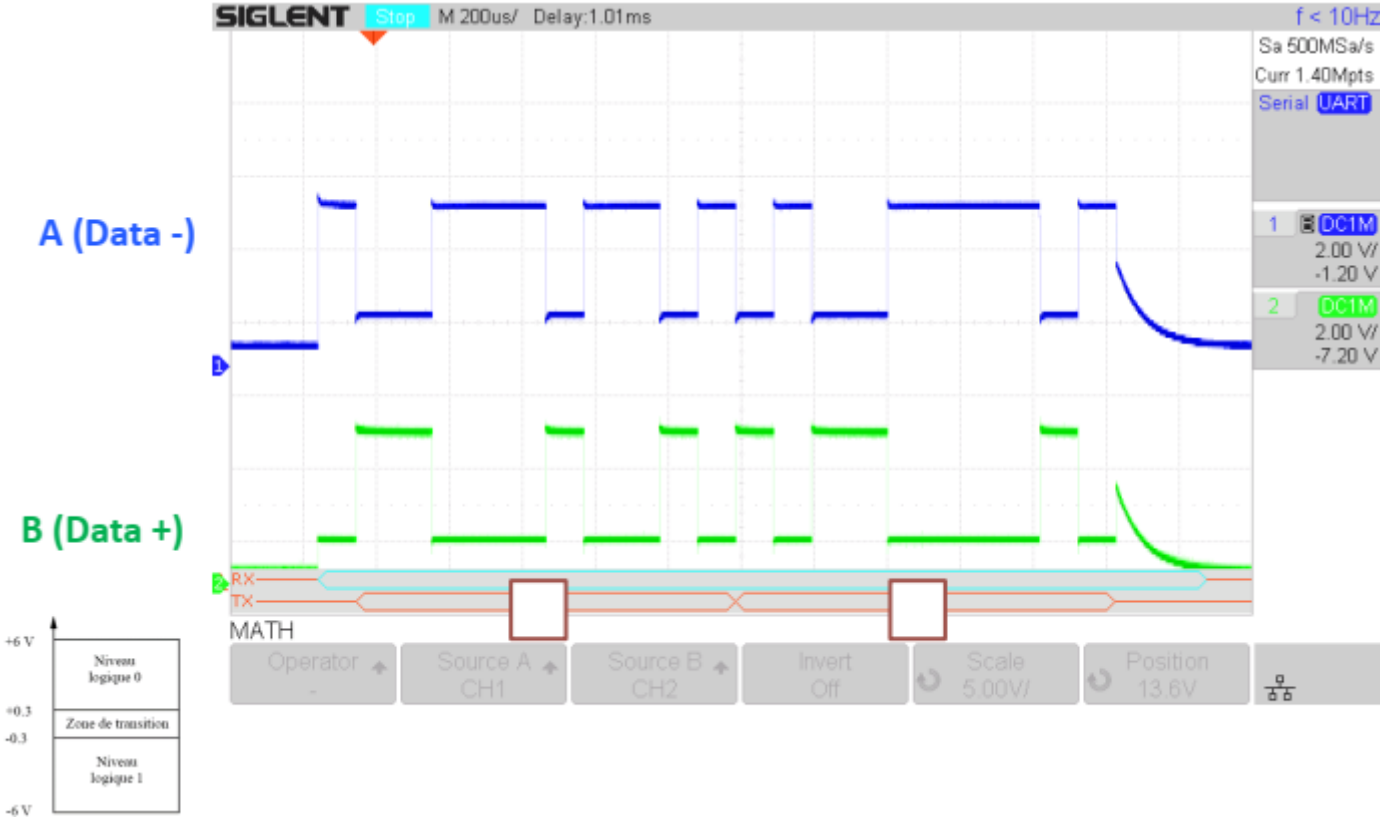
‘0’ logique = 0V



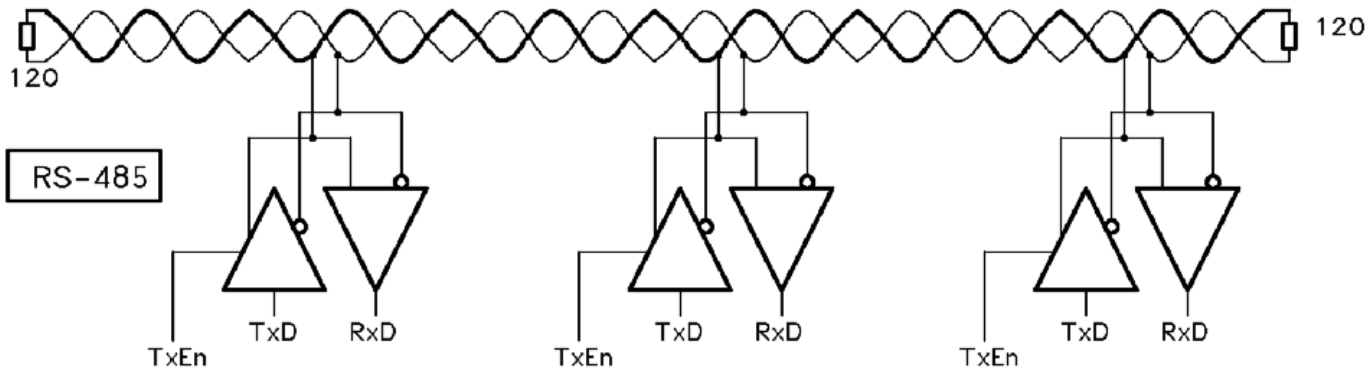

# Transmission série asynchrone RS485 vs RS232



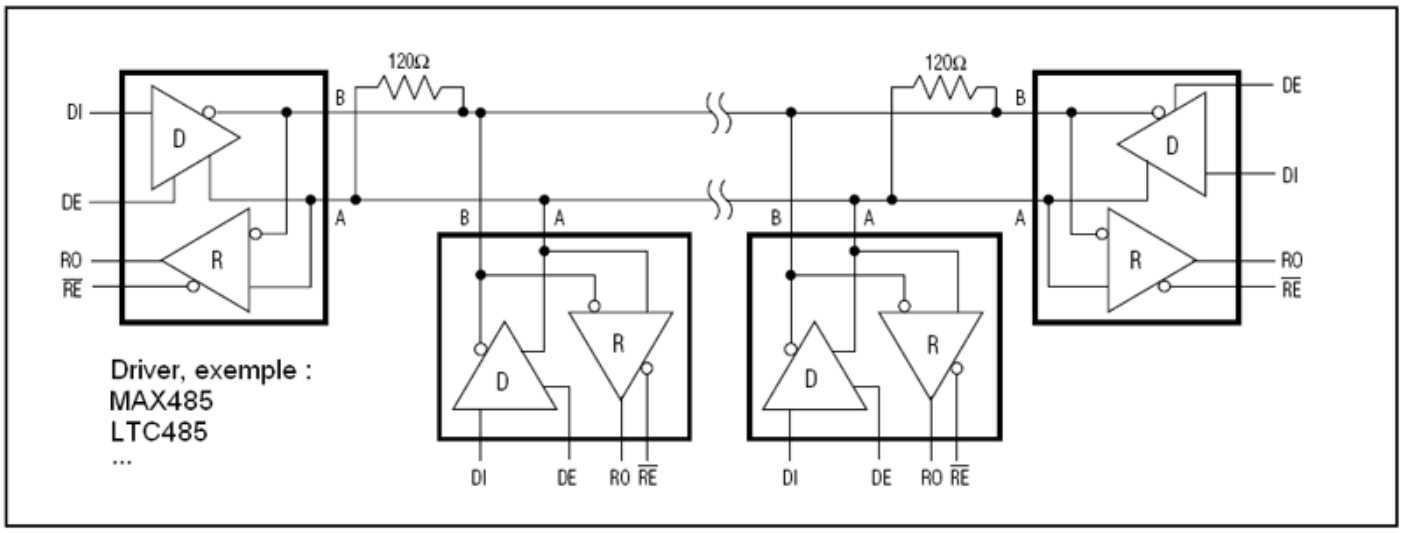

# Transmission série asynchrone RS485

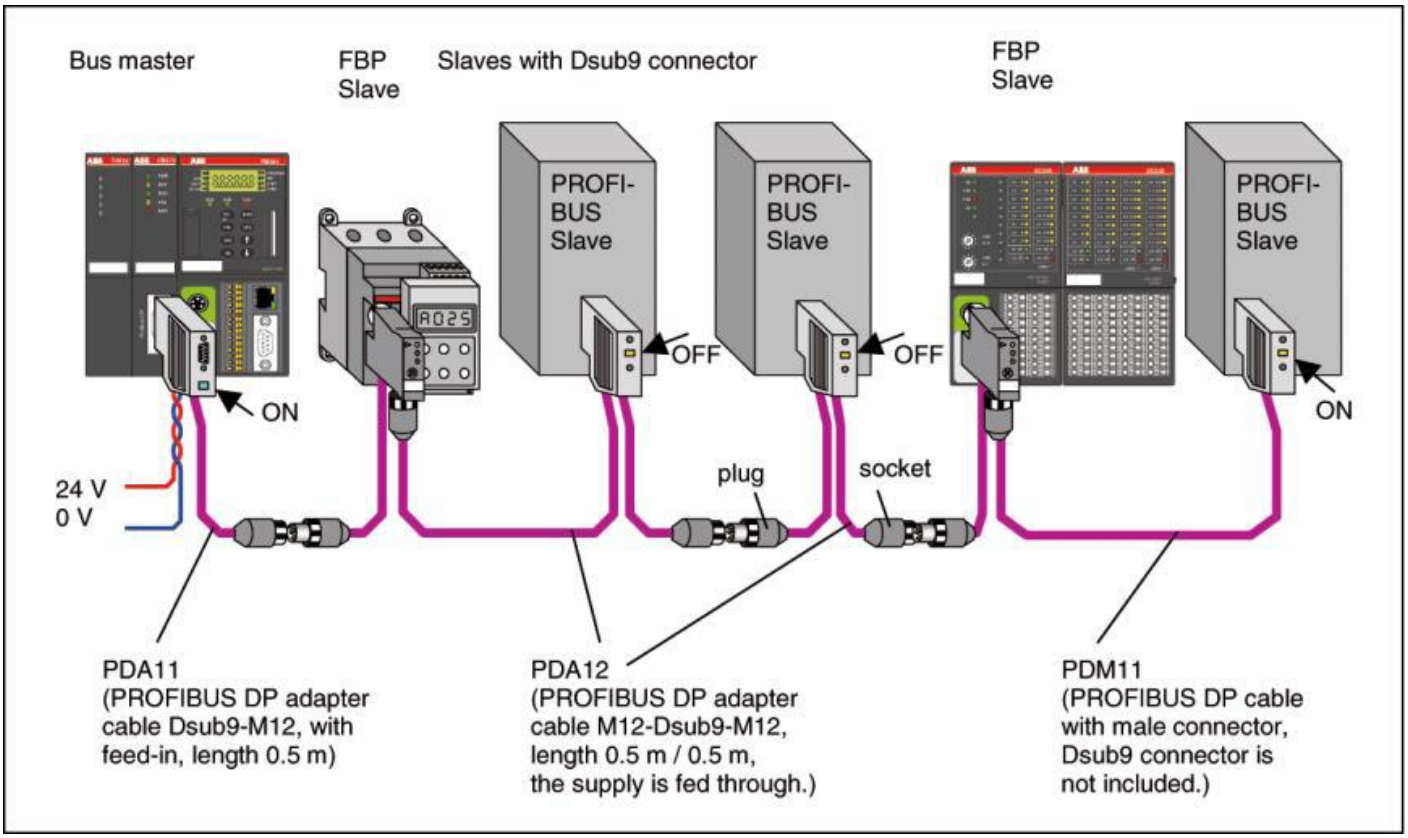


# Liaisons multipoints




Penser aux résistances de Terminaison de 120 Ω au début et à la fin de la liaison RS485.






# Half Duplex

Définition

- Liaison bidirectionnelle.
- 1 canal de transmission est partagé :
  - Il est utilisé dans un sens et dans l’autre.
  - Une règle doit définir comment gérer l’accès au média.
  - Moins cher, plus facile, mais plus lent.

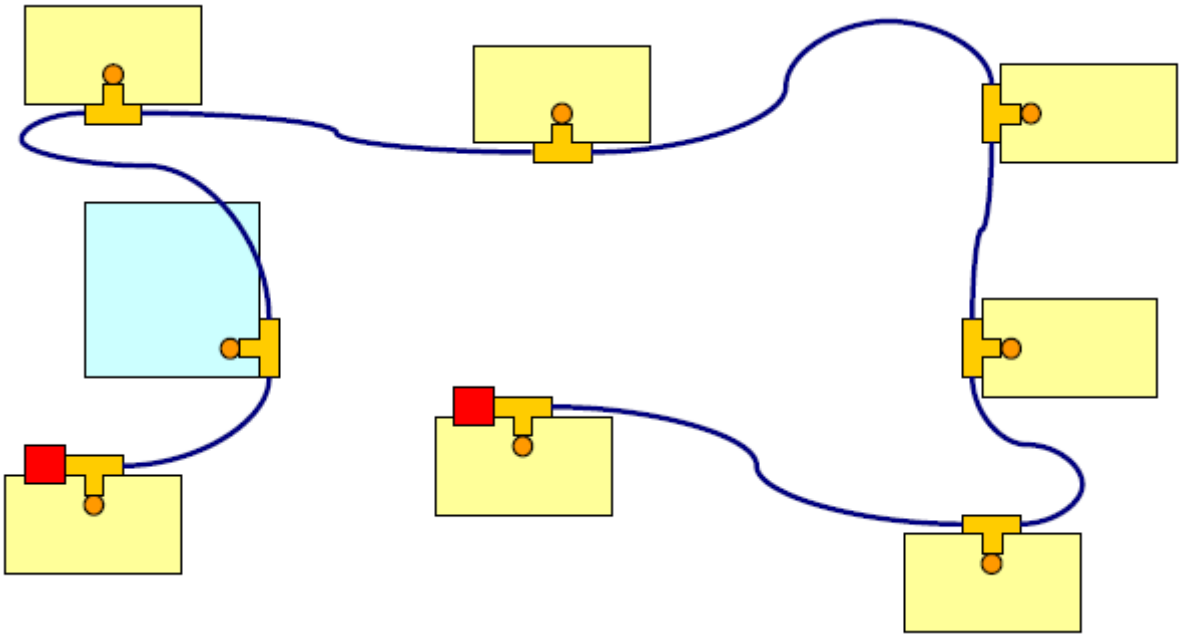
Exemple

- De nombreux bus de terrain, RS485,

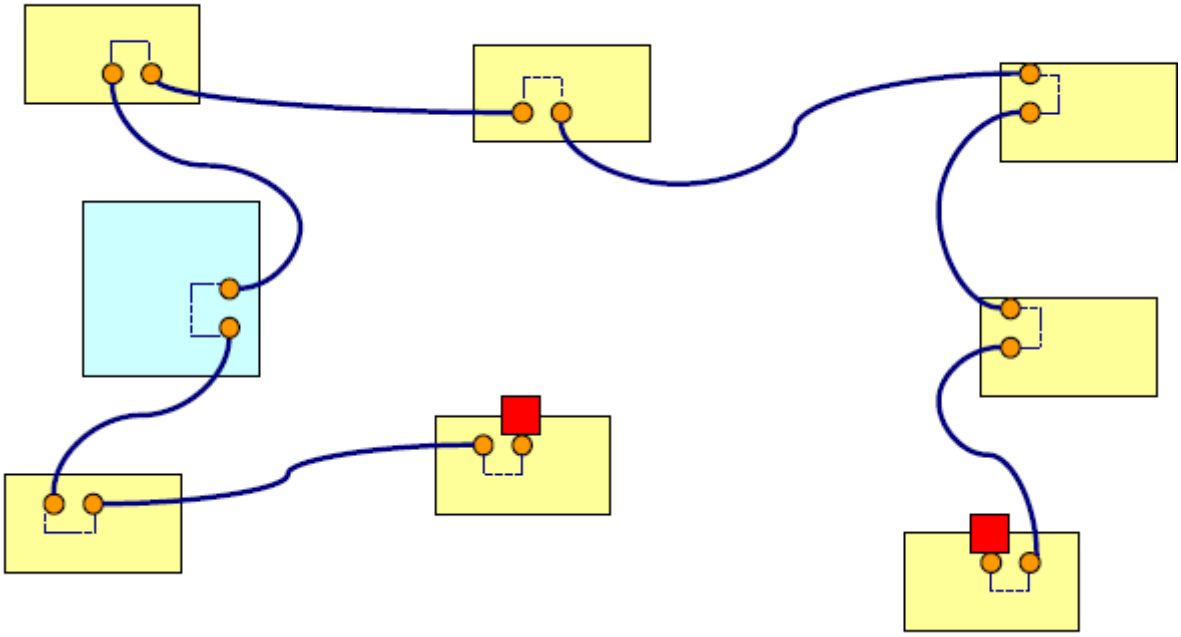
# Topologie : Bus

Principe

- Connections de toutes les stations sur un même câble
  - Toujours half duplex.
- 2 topologies selon les possibilités techniques
  - connexion en T "par prise vampire"



  - chaînage



Avantages

- Simplicité d’adjonction de stations.
- Fonctionne même en cas de panne d’une station.
- Transmission en diffusion ( broadcasting , multicasting
- Longueur de câble réduite.

Inconvénients

- 1 seule station peut émettre à la fois.
- Les résistances de terminaison sont externes (à câbler).
- Liaison en chaîne : échange d’appareil impossible sans arrêt du système.
- Liaison en T : coûts de connexion plus importants.

Exemples

- Connexion en prise vampire : ASi
- Profibus, Modbus

Source : Cours IUT Haguenau - Département GEII - Automatisme Spé. 4 - Réseaux Locaux Industriels - Philippe Celka, le 28.02.2022

Philippe Celka Copyright © 2025 CC Attribution-Non Commercial-Share Alike 4.0 International

# Introduction à Modbus

# Protocole Modbus

Image not found or type unknown
logo Modbus

# Introduction

Modbus est un protocole de communication **non propriétaire** créé par Modicon en 1979. Les spécifications du protocole sont données librement sur le site de la Modbus Organization. Ce consortium a été créé par Schneider suite au rachat de Modicon en 1997 pour promouvoir Mobdus auprès des fabricants et utilisateurs.

Modbus est très populaire dans les environnement industriels car c'est un protocole simple, facile à intégrer, efficace, fiable, **ouvert** et royalty-free ! Vous pouvez très facilement intégrer Modbus dans vos projets à base d'ESP32, Raspberry, STM32 …

Le protocole Modbus était à l'origine un protocole sur bus série (Modbus RTU). Il a évolué pour s'intégrer aux technologies TCP/IP quand Ethernet est monté en puissance. On le retrouve dans les domaines de:

- gestion technique des bâtiments
- systèmes de management de l'énergie
- processus complexes d'automatisation industrielle

C'est une couche applicative (niveau 7 OSI) qui se base sur les liaison séries ou sur les trames Ethernet et les couches TCP/IP.

**Stack de communication Modbus :**

Image not found or type unknown
Modbus Stack

On distingue les différents modes de communication :

- **Modbus TCP** : communication TCP/IP basée sur le modèle client/serveur
- **Modbus RTU** : transmission **série** asynchrone via **RS-485**, RS-232 ou RS-422.
- Modbus ASCII : similaire au protocole RTU, format sur 7 bit (utilisation très rare)

Nous débuterons l'analyse du protocole suivant la chronologie avec l'étude du **Mobdus RTU** (**R** emote **T**erminal **U**nit) sur liaison série.

# Modbus RTU

## Principe du protocole Master / Slave utilisé en Modbus Serial

La terminologie Master / Slave est remise en cause ces dernières années dans la communauté des développeurs et l’on évite de l’utiliser sur de nouveaux projets. Comme ces termes sont utilisés dans les spécifications officielles “Modbus Serial”, je continuerai des les employer sur cet exemple par cohérence avec les documentations.

**Principe de fonctionnement :**

- Seul un Master (au même moment) est connecté au bus, et un ou plusieur (247 maxi) Slaves sont également connecté sur le bus.
- Une communication Modbus est toujours initié par le Master. Les Slaves ne vont jamais transmettre de données sans requête du Master.
- Les Slaves ne peuvent pas communiquer entre eux.

Le Master peut initier une transaction avec le Slave suivant deux modes :

- **mode unicast** le Master s’adresse à un Slave individuel. Après réception de la requête et traitement de celle-ci, le Slave renvoie la réponse au Master. Dans ce mode, une transaction Modbus consiste en deux messages: la requête du Master (request) et la réponse du Slave (reply). Chaque Slave doit posseder une adresse unique (de 1 à 247) de manière à ce qu’il puisse être interrogé indépendament des autres Slaves.

  Modbus RTU unicast
  Image not found or type unknown

Le fait d’interroger les Slaves les uns à la suite des autres consiste à effectuer du “Polling”.

- **mode broadcast** le Master envoie **un message à l’ensemble des Slaves**. Les messages de broadcast sont forcément de type écriture. L’adresse 0 est reservée pour identifier un échange de type broadcast.

## Description du protocole

Le protocole Modbus définie un Protocol Data Unit (**PDU**) indépendant des couches de communication. Il s’agit de la structure du message de base :

Modbus PDU
Image not found or type unknown

Function Code représente le type d’ordre (lire, écrire) et les datas sont les paramètres de l’ordre (lire 4 registres mémoire depuis l’adresse 0x3214 par exemple).

Empaqueter le protocole Modbus sur un bus série ou Ethernet nécessite des champs additionels au PDU.

Modbus RTU PDU
Image not found or type unknown

Sur une liaison Modbus série, l’Address field contient uniquement l’adresse du Slave.

Le champ CRC contient un code de contrôle d’intégrité de message pour détecter les erreurs de transmission.

# Les règles d’adressage Modbus

Les adresses des appareils (devices) Modbus sont codés sur 1 octet (8 bits). Il y a donc 256 adresses possibles.

| 0 | From 1 to 247 | From 248 to 255 |
|---|---|---|
| Broadcast address | Slave individual addresses | Reserved |

- L’adresse 0 est réservée comme adresse de broadcast.
- Le Master Modbus n’a pas d’adresse spécifique. Seuls les Slaves doivent posséder une adresse qui doît être unique sur le bus série.
- Le fait que les spécifications Modbus indiquent qu’il est possible d’affecter des adresses comprises entre 1 et 247 ne veut pas forcément dire que tous les fabricants permettent cet interval (certains fabricants limitent les adresses de 1 à 100).

# Les types de données

Il y a deux types de données en Modbus, le bit et le Word (16 bits).

| | Type d’objet | Accès | Exemple |
|---|---|---|---|
| Discrete Input | bit | Read-Only | Entrée TOR, fin de course, contact auxilliaire de disjoncteurs, … |
| Coil | bit | Read-Write | Sortie TOR, bit interne, RAZ d’un compteur d’énergie, … |
| Input Register | Word (16 bits) | Read-Only | Entrée analogique, lecture d’un capteur, … |

| | Type d’objet | Accès | Exemple |
|---|---|---|---|
| Holding Register | Word (16 bits) | Read-Write | Sortie analogique, variable de programme (ex : temporisation, opérande d’un calcul,…) Valeur de paramétrage d’un équipement (ex: consigne de vitesse d’un variateur de fréquence,…) |

- Input et Input Register correspondent à des entrées. Ce sont des variables que l’on peut uniquement accéder en lecture (Read-Only).
- Coil (bobine) et Holding Register correspondent à des sorties que l’on peut forcer (write) mais également lire (read).

Un registre est codé sur 16 bits. Holding Register correspond ainsi à 16 Coil en mode Read-Write tandis que Input Register correspond à 16 entrées que l’on peut seulement acceder en lecture (Read-only).

**Rappel :**

- 1 Word = 2 bytes = 16 bits
- 1 Register est codé sur un Word soit 16 bits

# Les fonctions Modbus

Les “Function Code” correspondent aux types d’ordres, lire ou écrire par exemple, ainsi que le type d’accès (accès au niveau bit ou au niveau registre de 16 bits). Les fonctions sont identifiées par un code sur 8 bits qui peut être représenté en décimal ou en hexa.

### Bit access

| Code | Hex | Nom de fonction | Commentaire |
|---|---|---|---|
| 02 | 0x02 | Read Discrete Inputs | Physical Discrete Inputs |
| 01 | 0x01 | Read Coils | Internal Bits or Physical coils |
| 05 | 0x05 | Write Single Coil | Internal Bits or Physical coils |
| 15 | 0x0F | Write Multiple Coils | Internal Bits or Physical coils |

### 16-bit access (register)

| Code | Hex | Nom de fonction | Commentaire |
|---|---|---|---|
| 04 | 0x04 | Read Input Register | Physical Input Registers |

| Code | Hex | Nom de fonction | Commentaire |
|---|---|---|---|
| 03 | 0x03 | Read Holding Registers | Internal Registers or Physical Output Registers |
| 06 | 0x06 | Write Single Register | Internal Registers or Physical Output Registers |
| 16 | 0x10 | Write Multiple Registers | Internal Registers or Physical Output Registers |

Les tableaux ci-dessus ne sont pas exhaustif, il y a également des Function Code pour réaliser du diagnostique. Il faut savoir que les fabricants de matériel Modbus n'intègre pas forcément toutes les fonctions possibles. Les fonctions Modbus disponibles sont données dans la documentation technique du constructeur.

**Description d'une trame Modbus série**

Une communication Modbus série est définie par

- vitesse en bit/s ( 9600, 19200, 115200, autre )
- 1 bit de start
- 8 bits de données (LSB envoyé en premier)
- 1 bit de parité
- 1 ou 2 bit de stop

Classiquement, en Modbus RTU, c'est la parité paire (**Even**) qui est utilisée. Si l'on choisit de ne pas implémenter le contrôle de parité (**None**) il faut placer 2 bits de stop.

Une trame Modbus RTU est composée *a minima* de 4 octets et au maximun de 256 octets. Chaque octet (byte) qui compose une trame Modbus est codé de la manière suivante :

Modbus RTU frame
Image not found or type unknown

**Une trame Modbus RTU**

Une trame Modbus RTU est ainsi composée :

- 1 byte pour Slave Address
- 1 byte pour Function Code
- 0 à 252 byte pour Data
- 2 bytes pour le CRC

Modbus RTU Frame
Image not found or type unknown

La taille maximale d'une trame Modbus RTU est de 256 bytes.

Le CRC est calculé avec l'algo CRC-16-MODBUS.

**Acquisition d’une trame Modbus de type request**

Scope Frame Modbus RTU
Image not found or type unknown

Le décodage de trame Modbus intégré donne au format hexa la trame suivante :

`01 03 00 01 00 02 95 CB`

On en déduit :

- Slave Address : `01`
- Function Code : `03` -> Read Holding Register
- Data : `00 01 00 02`
- CRC : `95 CB`

Pour Data, suivant les caractéristiques de la fonction `03` Read Holding Register, les deux premiers bytes `00 01` corresponde à l’adresse de registre de départ et les deux suivants `00 02` correspondent aux nombre de registres que l’on souhaite lire à partir du registre de départ.

En résumé: la trame Modbus RTU suivante effectue la requête suivante -> Au Slave `01`, donne la valeur des `00 02` premiers registres à partir de l’adresse mémoire `00 01`.

# Branchement Modbus RTU en configuration 2 Wire

Le branchement Modbus RTU classique est le “2 Wire” en conformité avec le standard RS-485. Sur un “2W-Bus”, seul un driver à la fois a la possibilité de transmettre un message.

- LT : Line Terminator, c’est les résistance de terminaison (polarisation) du Bus. Elles font classiquement 120Ω120Ω ou 150Ω150Ω
- Les résistances de terminaison sont placées au début du bus et à la fin du bus.
- Balanced Pair : Paire de fils torsadés qui constituent le support de transmission.

Modbus Two 2 Wire
Image not found or type unknown

On parle de topologie 2 fils (2-Wire), mais on se rend compte sur le schéma, que finalement, 3 fils sont utilisés avec la masse (Common).

| Modbus Name | RS-485 Name | Autre Nom | Description |
|---|---|---|---|
| D1 | B | D+ ou Data+ | Transceiver Terminal 1 (V1>V0 for binary 1 [OFF] state) |
| D0 | A | D- ou Data- | Transceiver Terminal 0 (V0>V1 for binary 0 [ON] state) |

| Modbus Name | RS-485 Name | Autre Nom | Description |
|---|---|---|---|
| Common | C | 0v ou GND | Commun, Masse (0V) |

En RS-485, à 9600 bit/s sur une paire torsadée en AWG26, on arrive à une longueur de bus maximale de 1000 m!

Les résistances de polarisation (RPull−UpRPull−Up et RPull−DownRPull−Down) permettent de limiter le bruit sur le bus quand il n’y a pas de communication. Les valeurs de ces résistances sont comprises entre 450Ω450Ω et 650Ω650Ω.

**Remarques :** Il existe également des configurations de branchement en 4 fils (4-Wire) mais c’est rare.

# Connectique Modbus RTU

En Modbus RTU RS-485, trois types de connecteurs connecteurs sont souvent utilisés :

- bornier à visser (ou borne automatique)
- connecteur DB9
- connecteur RJ45

## Bornier à visser :

Sur le Wago Controller 100, la connexion se fait par un bornier automatique et utilise les abréviations D+ (D1 ou B) et D- (D0 ou A). L’abréviation GND est utilisée pour le commun (0V) et SH (Shield) pour une connexion au blindage.

Modbus Wago Controller 100
Image Modbus Wago Controller type unknown

## Connecteur DB9 :

L’automate PFC200 de chez Wago utilise une connectique DB9 qui permet de réaliser des liaisons RS-485 ou RS232. Pour le Modbus RTU, c’est la RS-485 qui est classiquement utilisée.

| PFC200 WAGO | Connecteur DB9 |
|---|---|
| Modbus Wago PFC200<br>Image Modbus Wago PFC type unknown | Modbus Wago DB9<br>Image Modbus Wago DB9 type unknown |

La documentation constructeur donne les informations suivantes pour la connectique DB9 du PFC200 en mode RS485.

| Contact | Signal RS-485 | Description |
|---|---|---|
| 1 | NC | Not assigned |

| Contact | Signal RS-485 | Description |
|---|---|---|
| 2 | NC | Not assigned |
| 3 | A (Tx/Rx+) | Transmitt/receive Data+ |
| 4 | NC | Not assigned |
| 5 | FB_GND | Ground |
| 6 | FB_5V | Power Supply |
| 7 | NC | Not assigned |
| 8 | B (Tx/Rx-) | Transmitt/receive Data- |
| 9 | NC | Not assigned |
| Housing | Shield | Shield |

On se rend compte que Wago ne respecte pas la norme Modbus dans ce produit ! Ils appellent A -> Data + et B -> Data - qui correspond à la dénomination Profibus de Siemens ! Si votre communication ne fonctionne pas, il suffit parfois d’inverser les fils A-B car le fabricant a mélangé la norme.

# Connecteur RJ45

Les fabricants utilisent aussi parfois un connecteur RJ45 pour les liaison RS-485 ! L’erreur est de croire que l’on peut connecter ce type d’appareils sur un switch ou sur le port RJ45 de votre PC. **NE LE FAITES PAS !**

Bien qu’il s’agisse d’un connecteur RJ45, il s’agit d’une liaison série qui est transportée et il faut donc l’associer à une interface série et non au port RJ45 de votre PC ou de votre switch ! Les fabricants adoptent parfois la connectique RJ45 car les câbles sont peu chers avec un branchement qui est facile et rapide.

Wago Current Sensor Modbus RTU

Image not found or type unknown

La documentation Wago donne l’association des broches du connecteur RJ45 :

| Pin | Function |
|---|---|
| 1 | Ub |
| 2 | Ub |
| 3 | n.c. |
| 4 | A (Data+) |
| 5 | B (Data-) |
| 6 | n.c. |

| Pin | Function |
|---|---|
| 7 | GND |
| 8 | GND |

Toujours la même erreur chez Wago. Ils appellent A -> Data+ et B -> Data- qui correspond à la dénomination Profibus de Siemens.

## Connexion RJ45 et DB9 selon spécifications Modbus

Modbus DB9 RJ45

Image not found or type unknown

| Pin on RJ45 | Pin on DB9 | Level of requirement | Modbus | RS-485 | Description |
|---|---|---|---|---|---|
| 3 | 3 | optional | PMC | - | Port Mode Control |
| 4 | 5 | **required** | D1 | B | Transceiver terminal 1, V1 Voltage (V1>V0 for binary 1 [OFF] state) |
| 5 | 9 | **required** | D0 | A | Transceiver terminal 0, V0 Voltage (V0>V1 for binary 0 [ON] state) |
| 7 | 2 | recommended | VP | - | Positive 5..24 Vdc Power Supply |
| 8 | 1 | **required** | Common | C | Signal and Power Supply Common |

On se rend compte que Wago n’a pas suivi les recommandations de câblage fixées par la Modbus Organization, de nombreux fabricants font de même. Quand il s’agit d’appareillages d’un même constructeur, cela ne pose pas de soucis, par contre, il faut parfois inverser les signaux A et B quand on mélange les appareillages de fabricants différents sur un même bus Modbus RTU. En Modbus TCP, comme c’est sur du câble Ethernet, on n’a pas ce problème.

# Exemple : Modbus RTU avec un capteur de Température et

# Humidité

Dans cet exemple, je vais connecter un capteur de température et d’humidité PKTH100B-CZ1 qui communique en Modbus RTU avec mon ordinateur portable.

Pour que le PC portable puisse communiquer en RS-485, je lui ajoute un convertisseur FTDI USB-RS485, ainsi qu’un Oscilloscope pour visualiser les trames Modbus-RTU (côté didactique)

| Capteur PKTH100B-CZ1 | FTDI USB-RS485 | Oscilloscope |
|---|---|---|
| PKTH100-1 Sensor<br>Image not found or type unknown | FTDI USB-RS485<br>Image not found or type unknown | Siglent SDS1202X-E<br>Image not found or type unknown |

L’analyse de la documentation du câble FTDI USB-RS485 nous donne les informations suivantes :

# Les câble USB-RS485

FTDI USB-RS485 colors
Image not found or type unknown

| Signal | Couleur de fil |
|---|---|
| GND | Noir |
| (A) Data - | Jaune |
| +5V | Rouge |
| R de 120Ω120Ω pin 1 | Brun |
| (B) Data + | Orange |
| R de 120Ω120Ω pin 2 | Vert |

# Le capteur

PKTH100-1 Sensor
Image not found or type unknown

| Terminals number | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Identifying | GND | VCC | B | A |
| Description | Power- | Power+ | RS485- | RS485+ |

On remarque que sur la documentation du capteur, le signal A est nommé RS485+ tandis que sur la document du convertisseur USB-RS485, le signal A est nommé Data - …

On va rester pragramatique et brancher le fil A (jaune) sur le bornier A (4) du capteur et le fil B (Orange) du convertisseur vers la borne B (3) du capteur. Si jamais cela ne fonctionne pas, il suffira d'inverser ;)

Les masses devant être communes, on branchera le fil GND (noir) du convertisseur à la borne GND (1) du capteur.

Pour alimenter le capteur, j'utilise une alimentation de laboratoire de 24Vdc. Pareil, je brancherai le +24Vdc de l'alimentation à la borne VCC (2) du capteur et le 0V de l'alimentation à la borne GND (1) du capteur.

Pour les résistances de terminaison de 120Ω120Ω, je fais le choix de ne pas les placer dans un premier temps car la longueur de bus est très faible.

# La manipulation

Manipulation Modbus Capteur Temperature

Image not found or type unknown

La documentation (en chinois) indique les paramètres suivants :

- Vitesse de transmission : 9600 bit/s
- 8 bits de données
- Parity : None
- 1 Stop bit (non respect de la norme)
- Slave Address (factory) : 1

La document indique également que la requête à envoyer est une fonction de type `03` Read Holding Resgister à l'adresse de Slave `1` et que l'on lit à partir du registre mémoire `0` un nombre de 2 registres.

La trame à envoyer avec le CRC est la suivante : `01 03 00 00 00 02 C4 0B`

J'utilise le logiciel QModMaster pour générer facilement la trame et et bien sur, cela ne fonctionne pas :(

> On va essayer d'inverser les fils A et B -> boum, ça fonctionne…bref

## Les différentes étapes de la configuration de qModMaster

On le numéro du port Com utilisé par le convertisseur USB-RS485 avec le gestionnaire de périphériques Windows. On remarque que dans mon cas, c'est le COM5 qui lui a été attribué. Cela nous permet de paramétrer la liaison série RTU dans QModMaster avec le bon numéro de Com et

l’on saisie également les paramètres de liaison du capteur de température (9600 bit/s 8bits de données 1 bit de stop et parity None)

| Gestionnaire de périphériques | Config Serial dans QModMaster |
| --- | --- |
| Image not found or type unknown | Image not found or type unknown |

Dans QModMaster, je choisis le Mode RTU, le Slave Address à 1, le Function Code à `0x03` pour Read Holding Register, le Start Address à 0 et Number of Coils (Registers) à 2.

Dans le Bus Monitor, on remarque que la trame de request vaut : `01 03 00 00 00 02 C4 0B` -> ce qui était demandé par la doc, donc on est OK !

La trame de réponse (reply) du capteurs vaut : `01 03 04 01 28 02 22 FA BE`

| QModMaster | Bus Monitor |
| --- | --- |
| Image not found or type unknown | Image not found or type unknown |

Le décodage du résultat est donné directement par QModMaster:

- Le premier registre vaut : `296` en décimal

- Le second registre vaut : `546` en décimal

La documentation du capteur indique que la valeur du premier registre correspond à la température multipliée par 10. On en déduit qu'il fait 29,6°C en cette journée d'août -> c'est OK

L'humidité multipliée par 10 est dans le second registre. On en déduit que l'humidité relative Hr=54.6%Hr=54.6% ce qui est conforme.

## Méthode de décodage à partir de la trame de réponse

La trame de réponse (reply) du capteurs vaut : `01 03 04 01 28 02 22 FA BE` . On peut décoder le contenu de la manière suivante :

- `01`: correspond à l'adresse du capteuur qui donne la réponse
- `03`: indique qu'il répond à une réquête de type 03 Read Holding Register
- `04`: c'est la valeur de la fonction 03 + 1 pour dire que tout c'est bien passé !
- `01 28`: c'est la valeur en hexa du contenu du premier registre avec `01` l'octet de poids fort et `28` l'octet de poids faible. Converti en décimal, on obtient `296`
- `02 22`: correspond à la valeur en hexa du second registre. Converti en décimal, on obtient `546`.
- `FA BE`: correspond au CRC de la trame de réponse.

## Capture des trames Modbus RTU avec l'oscilloscope

On peut observer la trame de request générée par QModMaster qui vaut `01 03 00 00 00 02 C4 0B`

Modbus Oscilloscope Frame
Image not found or type unknown

Et la trame de réponse du capteur qui vaut `01 03 04 01 28 02 22 FA BE`

Modbus Oscilloscope Frame
Image not found or type unknown

Source https://celka.fr/ocw/plc-control/modbus/intro-modbus/intro/

Philippe Celka Copyright © 2025 CC Attribution-Non Commercial-Share Alike 4.0 International

# Machine Learning LeRobot avec SO-ARM101 - Legacy

## Essai de déploiement sur Windows via WSL2, Docker, et Dev Container

> Pour l'instant pas de test satisfaisant pour l'exécution d'un modèle sur le vrai robot. Passer au WSL2 les ports USB où sont connectés les robots et caméras fait crasher le conteneur. Probablement que la communication série n'est pas supporté.

- Cloner https://github.com/huggingface/lerobot dans un conteneur WSL2, par exemple Ubuntu
- Depuis le conteneur Ubuntu, ouvrir un Terminal, se placer dans le répertoire cloné `cd ~/lerobot_devcontainer` , et lancer Visual Studio Code en tapant `code .`
- Ajouter un répertoire `~/lerobot_devcontainer/.devcontainer` et un fichier dedans `~/lerobot_devcontainer/.devcontainer/devcontainer.json` contenant :

  ```
  {
  "build": {
  // Path is relative to the devcontainer.json file.
  "dockerfile": "../docker/lerobot-gpu-dev/Dockerfile"
  }}
  ```

- Lancer la commande VSCode : Reopen in container

Tentatives pour faire passer le port série à WSL2 :

```
wsl --shutdown
winget install --interactive --exact dorssel.usbipd-win
usbipd list
wmic diskdrive list brief
wsl --mount \\.\PHYSICALDRIVE1
wsl.exe --version
wsl --mount \\.\PHYSICALDRIVE1 --bare
wsl --mount \\.\PHYSICALDRIVE1 --partition 2 --type ext4
wsl --shutdown
```

```
ipconfig
net localgroup docker-users "gauthier.hentz" /ADD
wsl --set-default ubuntu
wsl --shutdown
usbipd list
usbipd bind --busid 1-1
usbipd list
usbipd attach --wsl --busid 1-1
usbipd attach --wsl --busid 2-1
```

# 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
```

# Banc de machine learning avec SO-ARM101

## Assemblage et démarrage du SO-ARM101

### 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

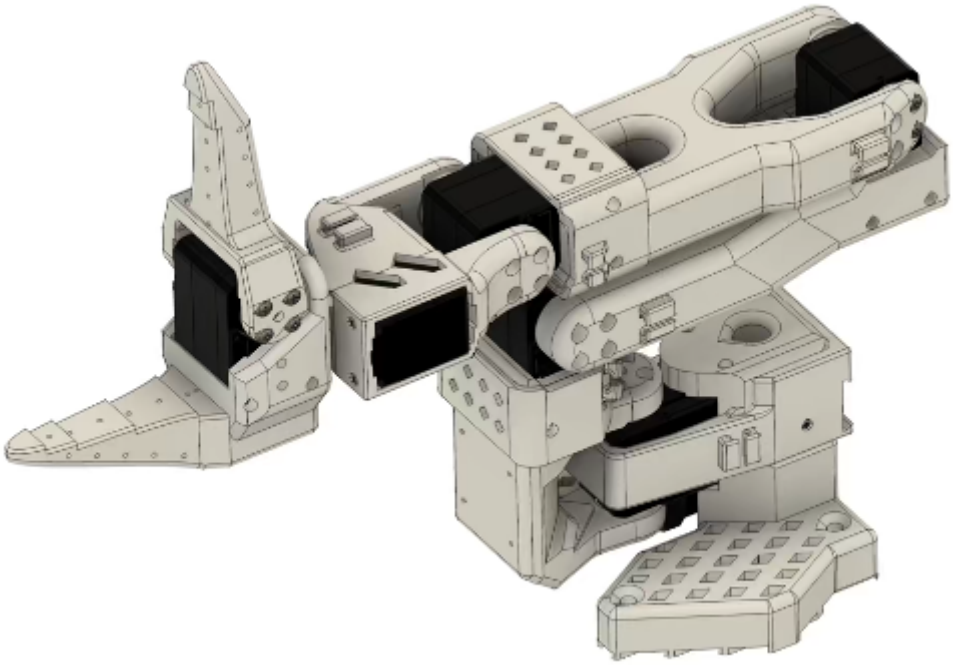
Une version 101 est sortie en 05/2025. Le Leader est plus simple à assembler, et plus besoin de démonter les servos pour enlever un engrenage et les rendre passifs. Il suffit d'acheter le kit de 6 servos avec 3 rapports de transmission différents :

- 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.0.0.757c2c3clU7uH3
- 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 modèle cinématique du SO-ARM10X



- 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 lancer la calibration LeRobot qui fixe l'Offset dans les servomoteurs, vous risquez de casser le robot

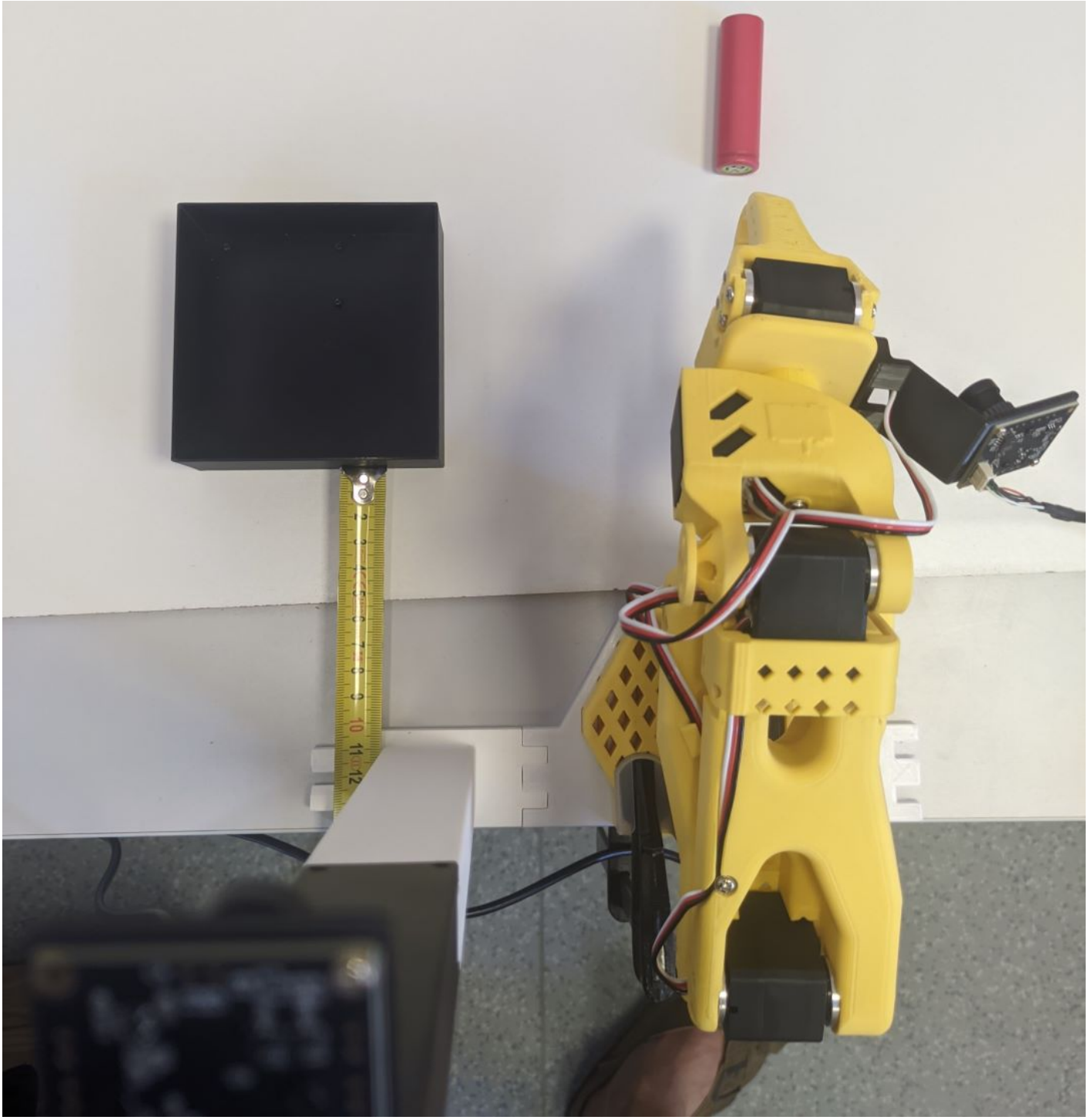
# Banc de Machine Learning LeRobot

## 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-ARM100/blob/main/Optional/Overhead_Cam_Mount_32x32_UVC_Module/README.md

## Calibration des caméras

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