Plugins DAP Pymodaq Branly 2026

pymodaq_plugins_raspberrypi3

Notice de Maintenance

Sommaire

Arborescence du code **p.2**
Côté plugin PyMoDAQ (src/pymodaq_plugins_raspberrypi3/) p.2
Côté serveur Raspberry (src_raspberry/) p.2

Communication ZMQ **p.3**
Trames JSON de référence p.3

Matériel (config.py) **p.4**

Ajouter un capteur ou un actionneur **p.5**
Côté serveur p.5
Côté plugin p.5

Conversion d'unités (à savoir) **p.5**
Supervision et journaux **p.5**
Terminal SSH p.5
Format de log serveur p.5

Points ouverts / pièges connus **p.6**

Dictionnaire des messages d'erreur (extrait Link_PMQ.py) **p.7**

Versionnage et publication **p.8**

Arborescence du code

Côté plugin PyMoDAQ (src/pymodaq_plugins_raspberrypi3/)

```
daq_move_plugins/daq_move_MoveRaspPi3.py   # actionneur
daq_viewer_plugins/plugins_0D/daq_0Dviewer_ViewRaspPi3.py
# détecteur 0D
hardware/Link_PMQ.py # classe ZMQLink (cœur communication)
# base commune scan + sélection composants
hardware/Config_Components.py
resources/config_template.toml  # modèle de config utilisateur
utils.py   # charge config_raspberrypi3.toml
```

Côté serveur Raspberry (src_raspberry/)

```
main.py
# point d'entrée : CProtocolHandler + CZmqServer (port 5555)
connexion.py
# CZmqServer (socket ROUTER) + build_sensor_map + monitor
protocol_I2C.py # CProtocolHandler : routage des requêtes JSON
sensors.py # drivers I2C + DRIVER_REGISTRY
actuators.py    # CActuatorManager (PWM via pigpio)
scanner.py # scan_bus() : détection des adresses I2C
config.py
# topologie matérielle (pins, adresses, bornes, fréquences)
safety_monitor.py
# boucle sécurité surchauffe (voir §8 — NON branché)
```

Communication ZMQ

- **Plugin = client DEALER** (Link_PMQ.py, classe ZMQLink). Chaque module reçoit un **identifiant unique** (uuid.uuid4()) injecté dans zmq.IDENTITY. **C'est ce qui empêche les données de se mélanger entre modules** (bug historique : sans uuid, ZMQ réutilisait le même identifiant et le daq_move récupérait parfois les données du daq_view).
- **Serveur = ROUTER** (connexion.py, classe CZmqServer), bind tcp://*:5555. Gère les formats **REQ** (3 frames avec délimiteur) et **DEALER** (2 frames). Un thread monitor_connections journalise connexions/déconnexions.
- Sérialisation **JSON UTF-8**.

Trames JSON de référence

Type	Requête	Réponse
Aquisition	{"type":"AQ","register":"add","add":i2c}	{"state":"ACK"/"ERROR","value":val /"msg"}
Multi Acquisition	{"type":"AQ-MULTI","components":[...] }	{"state":...,"value":[v1,v2,...]}
Pilotage	{"type":"PI","register":"add","add":i2c," value":v}	{"state":...,"value":nouvelle_val}

PI-MULTI est aussi prévu côté serveur (pilotage groupé). Le format JSON facilite l'ajout de nouveaux types : étendre CProtocolHandler (serveur) **et** ZMQLink (plugin). Alternative au transport : **LECO** (mentionnée par le CETHIL) peut remplacer PyZMQ.

Matériel (config.py)

Élément	Valeur Exemple
Bus I2C	I2C_BUS_ID = 1
Ventilateur	GPIO 18, PWM **25000 Hz**
Résistance (MOSFET)	GPIO 23, PWM **100 Hz**
AHT10 (rh_sortie)	0x38
TMP102	0x48 t_resistance, 0x49 t_dissipateur, 0x4A t_entree, 0x4B t_sortie
EMC2101 (T_emc)	0x4C

Actionneurs : bornes min=0, max=255. Protections de puissance : diode de roue libre + condensateur de découplage sur l'étage 15 V.

Ajouter un capteur ou un actionneur

Côté serveur

1. Créer une classe driver dans sensors.py (méthode ReadValue).
2. L'enregistrer dans DRIVER_REGISTRY.
3. Ajouter l'adresse + driver dans SENSORS_CONFIG (config.py). Le scan instancie automatiquement le bon driver (aucun if/else dans le cœur).

Côté plugin

Ajouter le composant dans config_raspberrypi3.toml (title/name uniques), recréer le preset.

Conversion d'unités (à savoir)

PyMoDAQ applique une conversion d'unité non désirée sur le pilotage. Contournement dans daq_move_MoveRaspPi3.py : les bornes sont multipliées par 100 (min/max ×100) et move_value divise la valeur par 100 pour annuler la conversion. À garder en tête si on retouche les bornes ou les unités.

Supervision et journaux

Terminal SSH

```
sudo systemctl status pilotage  # état
journalctl -u pilotage -f  # logs temps réel
(Ctrl+C pour quitter)
sudo i2cdetect -y 1  # capteurs présents
```

Format de log serveur

HH:MM:SS [INFO] connexion — NOUVEAU CLIENT : <ip> puis trames.

Points ouverts / pièges connus

- **safety_monitor.py non branché.** La boucle de sécurité surchauffe (force le ventilateur à 255 si T ≥ seuil, avec hystérésis) existe mais **main.py ne l'importe ni ne la lance**. De plus, elle référence config.SAFETY_SENSOR_ADDR, SAFETY_TEMP_MAX, SAFETY_HYSTERESIS, SAFETY_FAN_PIN qui **ne sont pas définis** dans config.py. Pour l'activer : définir ces constantes puis démarrer le thread dans main.py.
- **IP par défaut du template** : config_template.toml livre address_Rasp = '192.158.235.2' (probable coquille pour 192.168.…). Vérifier/corriger lors de la copie vers config_raspberrypi3.toml.
- **Import relatif** : hardware/Link_PMQ.py doit être importé en chemin relatif, sinon les modules n'apparaissent pas dans le Dashboard.
- **Nomenclature** : le paquet doit s'appeler pymodaq_plugins_xxx (sans underscores parasites), d'où pymodaq_plugins_raspberrypi3.

Dictionnaire des messages d'erreur (extrait Link_PMQ.py)

Message	Cause	Résolution
ERROR - ip address not set	IP non renseignée à open()	Renseigner l'IP dans le preset
ERROR : input type incorrect, dict required	Réponse non-dictionnaire (scan/AQ/PI)	Vérifier que le serveur renvoie bien un dict
ERROR : incorrect frame ACTUATOR/DETECTOR	Sous-dict scan incomplet (8 / 5 champs)	Vérifier le format renvoyé par le serveur
ERROR : incorrect frame, too much list or not enough	Réponse scan ≠ 2 listes attendues	Vérifier le format de trame
ERROR: hardware should have an address or a pin	Ni adresse ni pin fourni	Fournir l'un des deux
ERROR: only one of address or pin should be given, not both	Adresse **et** pin fournis	N'en fournir qu'un
ERROR - You need to choose at least one component in the config file (daq_0Dviewer_ViewRaspPi3.py)	Aucun composant coché	Cocher ≥ 1 composant dans le module

Versionnage et publication

- Versions v0 → v5 (v0 analyse/UML ; v1 fonctionnel minimal ; v2 IHM ; v3 sauvegarde + sélection composants ; v4 adaptable/extensible ; v5 documenté).
- Dépôt GitLab pymodaq-plugins-raspberrypi3 (groupe projet_pymodaq_2026), branches par version, merge sur main.
- pyproject.toml : pymodaq>=5.0.0, pyzmq>=27.0.0. Publication PyPI possible.