

# Fonctionnement

## Simulateur MQTT — Documentation

“ **parc-stub** est un simulateur de capteurs IoT pour le projet SIGACS.  
Il publie des données de serre sur le broker MQTT afin de tester le pipeline complet sans matériel réel.

### Aperçu de l'architecture

```
????????????????????????????????????????????????????????????????????
?  Navigateur (votre PC)                                                    ?
?  http://<serveur>:5001  ?  Web UI                                          ?
????????????????????????????????????????????????????????????????????
?  REST /api/*                                                                ?
????????????????????????????????????????????????????????????????????
?  Docker · parc-stub                                                       ?
?  ??????????????????????  contrôle  ?????????????????????????????????? ?
?  ?  Flask API  ?  ???????????????  ?  Thread publisher  ?  ?
?  ?  app.py      ?                ?  stub.py              ?  ?
?  ??????????????????????          ?????????????????????????????????? ?
?                                     ?                               ?
?  /data/config.json  ??? persistence ??????????                          ?
????????????????????????????????????????????????????????????????????
?  MQTT mTLS :8883                                                            ?
????????????????????????????????????????????????????????????????????
?  Docker · mosquitto                                                       ?
????????????????????????????????????????????????????????????????????
?                                     ?                               ?
????????????????????????????????????????????????????????????????????
?  Docker · bridge.py  ?  MySQL (parc)                                       ?
????????????????????????????????????????????????????????????????????
```

### Données simulées — capteurs

Le stub publie trois capteurs en **onde sinusoïdale** (cycle de 2 minutes).  
Chaque bac possède un déphasage aléatoire — les courbes diffèrent entre bacs.

Capteur	Clé JSON	Centre	Amplitude	Unité
Humidité ambiante	humiditeAmbiante	60	±20	%
Humidité du sol	humiditeSol	55	±15	%
Température ambiante	temperatureAmbiante	22	±5	°C

Les données capteurs ne sont publiées **que si le contrôleur de la serre est connecté**.

## Topic et payload capteurs

```
serre/<numero_serre>/bac/<numero_bac>
```

```
{"humiditeAmbiante": 67.4, "humiditeSol": 52.1, "temperatureAmbiante": 24.3}
```

## Données simulées — contrôleurs (M5Stack)

Chaque serre est associée à un contrôleur identifié par son adresse MAC.

Événement	Topic	Sens
Connexion	controleur/<MAC>	Stub → Broker
Déconnexion (LWT)	controleur/<MAC>/disconnect	Stub → Broker (retain)

## Payload de connexion

```
{  
  "topic": "controleur/24:D7:EB:38:DC:38",  
  "ip": "192.168.1.101",  
  "status": true  
}
```

## Payload de déconnexion (LWT)

Payload vide "", topic retain `controleur/<MAC>/disconnect`.

# Injection d'erreurs

Le taux d'erreur capteurs (0-100 %) déclenche aléatoirement l'un de ces 4 types :

Type	Payload envoyé	Erreur attendue côté bridge
out_of_range	<code>{"humiditeAmbiante": 150.0}</code>	VALUE_OUT_OF_RANGE
unknown_sensor	<code>{"co2Level": 400.0}</code>	UNKNOWN_SENSOR
malformed	<code>\xff\xfe ...</code> (non-UTF-8)	INVALID_ENCODING
invalid_value	<code>{"temperatureAmbiante": "hot"}</code>	INVALID_VALUE

## Web UI

Accessible sur `http://<IP_serveur>:5001`

Section	Rôle
<b>Contrôle global</b>	Démarrer / arrêter le stub, régler l'intervalle de publication et le taux d'erreurs capteurs
<b>Serres &amp; Contrôleurs</b>	Ajouter / supprimer des serres ; configurer MAC, IP, taux de déconnexion aléatoire ; connecter / déconnecter manuellement
<b>Journal</b>	Affichage en direct — vert = OK, orange = erreur capteur, bleu = connexion, rouge = déconnexion

## Couleurs du journal

Couleur	Signification
☐ Vert	Message capteur nominal
☐ Orange	Message capteur invalide
☐ Bleu	Connexion contrôleur
☐ Rouge	Déconnexion contrôleur (LWT)

“ Les numéros de serre et bac **doivent correspondre** aux colonnes `numero` dans la base de données, sinon le bridge logge `BAC_NOT_FOUND`.

# Persistance de la configuration

La configuration (serres, bacs, MAC, IP, intervalle, taux d'erreur) est sauvegardée automatiquement dans `/data/config.json` à chaque modification.

- Survit aux **refreshs** de l'interface
- Survit aux **redémarrages** du conteneur Docker
- L'état `connected` est toujours remis à `false` au redémarrage

Le fichier est stocké dans le volume Docker nommé `parc-stub-data`.

## API REST

Méthode	Route	Description
GET	<code>/api/state</code>	État complet (running, config, serres, log)
POST	<code>/api/start</code>	Démarre la publication
POST	<code>/api/stop</code>	Arrête la publication
POST	<code>/api/config</code>	Modifie <code>interval</code> (s) et <code>bad_rate</code> (0-1)
POST	<code>/api/serres</code>	Remplace la liste complète des serres/bacs
POST	<code>/api/serre/&lt;n&gt;/connect</code>	Connecte manuellement le contrôleur de la serre n°
POST	<code>/api/serre/&lt;n&gt;/disconnect</code>	Déconnecte manuellement le contrôleur de la serre n°

## Exemples

```
# Changer l'intervalle et le taux d'erreur
curl -X POST http://<serveur>:5001/api/config \
  -H "Content-Type: application/json" \
  -d '{"interval": 5, "bad_rate": 0.1}'

# Connecter manuellement la serre 1
curl -X POST http://<serveur>:5001/api/serre/1/connect

# Déconnecter manuellement la serre 2
curl -X POST http://<serveur>:5001/api/serre/2/disconnect

# Remplacer la configuration des serres
curl -X POST http://<serveur>:5001/api/serres \
```

```
-H "Content-Type: application/json" \  
-d  
'[{"numero":1,"mac":"24:D7:EB:38:DC:38","ip":"192.168.1.101","disconnect_rate":0.05,"bacs":[{"numero":1},{ "numero":2}]}]'
```

# Certificats mTLS

Le stub utilise un certificat client dédié généré par `setup.sh` :

```
mosquitto/stub-certs/  
??? stub.crt    ? certificat client (CN=parc-stub, 720 jours)  
??? stub.key    ? clé privée RSA 2048 bits (chmod 600)
```

Signé par la même CA que le reste du projet — aucun changement de config Mosquitto nécessaire.

“ **Erreur fréquente** : `[Errno 21] Is a directory` — un des chemins de certificat dans le volume Docker pointe vers un dossier. Vérifier avec `docker inspect parc-stub | grep -A 20 Mounts`.

# Commandes utiles

```
# Logs en direct  
docker logs -f parc-stub  
  
# Redémarrer sans rebuild  
docker compose -f parc-stub/docker-compose.stub.yml restart  
  
# Reconstruire après modification des sources  
docker compose -f parc-stub/docker-compose.stub.yml up -d --build  
  
# Arrêter et supprimer le conteneur  
docker compose -f parc-stub/docker-compose.stub.yml down  
  
# Inspecter les volumes montés  
docker inspect parc-stub | grep -A 20 Mounts  
  
# Lire la config persistée  
docker exec parc-stub cat /data/config.json
```

# Fichiers du projet

```
parc/
??? setup.sh           ? configuration initiale (certificats + docker-compose)
??? mosquito/
?   ??? stub-certs/
?     ??? stub.crt     ? certificat client du stub
?     ??? stub.key     ? clé privée (non committée)
??? parc-stub/
    ??? Dockerfile
    ??? docker-compose.stub.yml ? service + volume parc-stub-data
    ??? requirements.txt
    ??? stub.py         ? logique de publication + persistance
    ??? app.py          ? API Flask
    ??? templates/
        ??? index.html ? interface Web
```

---

*Projet SIGACS — BTS CIEL · Saint Joseph LaSalle · Lorient*  
*HERVOUET Clément · BANCQUART Alan · LE GOUALEC Titouan*

---

Revision #1

Created 2026-05-21 14:05:20 UTC by Clément

Updated 2026-05-21 14:06:34 UTC by Clément