TRACKER SST — Manuel Technique

Backtracking & MQTT — Documentation d'intégration avancée

25 septembre 2026

Sommaire

Introduction

Partie 1 — Backtracking : documentation technique

Partie 2 — MQTT : documentation du protocole

Introduction

Ce document complète le *Manuel Utilisateur TRACKER SST* déjà livré. Il ne s'adresse pas à l'utilisateur final mais à un **installateur ou intégrateur technique** : il documente, en détail, le fonctionnement interne de deux sous-systèmes du firmware — la fonction **Backtracking** (anti-ombrage entre plusieurs trackers alignés) et le **protocole MQTT** (communication à distance, tableau de bord web, intégrations tierces).

> Comme le manuel utilisateur, ce document a été rédigé directement à partir du code source réel du firmware, ligne par ligne — chaque comportement décrit ici a été vérifié dans le code, et plusieurs affirmations-clés ont été recontrôlées une seconde fois. Ce document est délibérément plus technique et plus cru que le manuel utilisateur : il inclut des structures de données, des extraits de code, des noms de fonctions, et surtout les limites, approximations et points non vérifiés sur matériel réel que le code documente lui-même. Rien n'est enjolivé : un algorithme plus simple que son nom ne le laisse penser, une valeur de seuil non calibrée sur site, ou une fonction laissée inactive sont signalés explicitement plutôt que passés sous silence.

Par précaution de sécurité, la valeur littérale du secret partagé `BACKTRACK_LICENSE_SECRET` n'est reproduite nulle part dans ce document — seuls son rôle, son format et la façon de le renouveler sont documentés.

Partie 1 — Backtracking : documentation technique

1.0 Vue d'ensemble architecturale

Le Backtracking (nom de code interne « BRC » dans les commentaires du firmware) est une **correction appliquée par-dessus** le mode de suivi actif (Auto / Secours / CRB), et non un mode de suivi supplémentaire à part entière. Il agit **uniquement sur l'axe de tilt** (jamais sur l'azimut) et n'a de sens qu'en axe double (le réglage Admin « Axe de suivi » sur Double).

Le transport entre trackers utilise la radio **ESP-NOW native de l'ESP32**, traitée sur le même cœur que le tracking (« core0 ») pour l'essentiel du calcul, avec l'initialisation radio elle-même déplacée sur le second cœur (« core1 », partagé avec la couche réseau/MQTT décrite en Partie 2) — voir §1.4. Le fichier porteur de la quasi-totalité de la logique est `TRACKER_base_55_57_03_SD_Backtracking_Courbe.ino` ; le menu Admin et les écrans dédiés sont dans `TRACKER_base_55_57_06_Menus_Admin.ino`.

1.1 Structures de données

`BacktrackPoint` — un point de la courbe locale de secours du tracker (8 octets) :

```
struct BacktrackPoint
{
    uint16_t minuteOfDay;   // minute du jour (0-1439)
    int16_t  azDeg;         // azimut courant en degrés
    int16_t  tiltDeg;       // tilt courant en degrés
    int16_t  ldrRaw;        // moyenne brute des 4 LDR (E/O/H/B)
};
```

`BacktrackLastDayBackup` — la courbe complète propre à ce tracker (distincte de la courbe du mode CRB) :

```
struct BacktrackLastDayBackup
{
    uint16_t       year;
    uint8_t        month;
    uint8_t        day;
    uint16_t       pointCount;
    BacktrackPoint points[268];   // BACKTRACK_LASTDAY_POINTS_MAX
};
```

Un point est enregistré uniquement les 1er et 15 de chaque mois, à la cadence réglée (« Cadence écriture »), et seulement lorsque les LDR/moteurs sont stables depuis un court délai.

Messages ESP-NOW — quatre structures distinguées **uniquement par leur taille exacte** à la réception (pas d'en-tête dédié) :

- **`BacktrackEspNowMsg`** (télémétrie / correction / configuration de champ / auto-déclaration HS) — un seul format pour les deux sens de communication : position émetteur, position cible, minute du jour, azimut, tilt, LDR brut, et un champ `tiltCorrectionDeg` recyclé selon le type de message.
- **`BacktrackDashEspNowMsg`** — passerelle dashboard/MQTT, miroir de 18 champs de télémétrie étendue (mode, sécurité, LDR, vent, batterie, moteurs, état I2C, bypass maintenance…).
- **`BacktrackCmdEspNowMsg`** — relais d'une commande MQTT vers un Esclave silencieux : position cible + `payload[128]` (texte JSON brut) + indicateur d'autorisation déjà vérifié côté Maître.
- **`BacktrackMaintAuthEspNowMsg`** — diffusion nocturne des badges NFC autorisés pour le bypass de maintenance tilt.

`BacktrackPeerInfo` / `BacktrackPeerTable` — table interne tenue uniquement par le Maître, **exclusivement en RAM** (jamais persistée en EEPROM ni SD — un redémarrage du Maître efface toute connaissance de la flotte, y compris les déclarations HS manuelles). Plafonnée à 64 trackers (`BACKTRACK_PEER_MAX`) ; au-delà, les trackers en surnombre sont simplement ignorés du calcul plutôt que de risquer un dépassement mémoire.

1.2 Stockage SD et EEPROM

Donnée	Emplacement	Format
Courbe locale (`BacktrackLastDay`)	`/backtrack_tracker/lastday.bin` (carte SD)	En-tête magic + CRC32 + longueur, puis la structure binaire brute
État de licence chiffré	`/backtrack/license.bin` (carte SD)	Bloc de 16 octets chiffré AES-128 (voir §1.7)
Réglages Admin (rôle, espacements, comptages, hauteur, pas de correction, position)	Bloc EEPROM principal, 37 octets ajoutés en fin de bloc	Champs `BacktrackEnabled/Role/SpacingX/SpacingY/CountX/CountY/Height/TiltStep/PosX/PosY`
Jeton distant (`BacktrackRemoteToken`)	Bloc EEPROM principal, 33 octets	Chaîne texte, vide par défaut
Copie de secours (réglages + licence)	EEPROM DS1307 (puce dédiée à l'horloge, bloc `"NET1"` séparé)	Filet de secours indépendant, survit à un changement de version du bloc principal

Bornes de sécurité appliquées au chargement, valeur par défaut restaurée hors plage : Rôle ∈ {Esclave, Maître} sinon Esclave ; Espacements 1-400 m sinon 5 ; Nombres de trackers 1-64 sinon 1 ; Hauteur 1-20 m sinon 2 ; Pas de correction 2-10° (pair) sinon 2 ; Position re-bornée automatiquement si un comptage réduit la rend invalide.

Point notable : les anciennes fonctions de repli vers une seconde carte SD physique (architecture historique à deux puces) **n'ont plus d'effet** depuis la fusion sur puce unique S3 — elles renvoient simplement un échec propre si la carte SD locale est absente, sans tenter de basculer ailleurs.

1.3 Rôles Maître / Esclave

Réglage Admin > Backtracking > « Rôle (Esclave/Maître) », un seul Maître prévu par champ (installé au centre ; limite de conception : champ jusqu'à environ 400 × 400 m). Un Maître reste **aussi** un tracker normal du champ : il émet sa propre télémétrie et applique ses propres corrections exactement comme un Esclave — seul le calcul de correction lui est réservé.

Un Esclave :

- émet sa télémétrie (position, azimut, tilt, LDR moyen) en broadcast périodique ;
- ne calcule jamais de correction lui-même, il reçoit et applique une correction ciblée envoyée par le Maître ;
- **cesse d'ouvrir toute connexion réseau individuelle** (MQTT, météo, Tuya) dès que le Backtracking est actif — seul le Wi-Fi en mode station reste actif, nécessaire à ESP-NOW ; le Maître devient l'unique passerelle réseau du champ ;
- reçoit et applique automatiquement les diffusions de configuration de champ (espacements, comptages) envoyées par le Maître, sans jamais avoir à les saisir manuellement.

Un Maître :

- calcule seul la correction d'ombrage à partir de la télémétrie de tous les trackers connus (table interne) ;
- envoie les corrections ciblées (et applique directement en local s'il est lui-même la cible) ;
- diffuse la configuration de champ à chaque début de balayage dashboard ;
- interroge le champ **rangée par rangée** pour le tableau de bord (jamais tout le champ en une fois, afin de limiter le nombre de réponses radio simultanées) ;
- sert de passerelle MQTT unique pour tout le champ et relaie les commandes reçues vers l'Esclave concerné en ESP-NOW.

1.4 Communication ESP-NOW entre trackers

Initialisation, effectuée une fois au démarrage (indépendamment de l'activation du Backtracking — la radio reste prête même désactivée) :

```
WiFi.mode(WIFI_STA);
esp_now_init();
esp_now_register_recv_cb(OnBacktrackEspNowRecv);
memset(peer.peer_addr, 0xFF, 6);   // FF:FF:FF:FF:FF:FF — broadcast
peer.channel = 0;
peer.encrypt = false;
esp_now_add_peer(&peer);
```

Adressage exclusivement en broadcast, jamais de peering MAC individuel par tracker — choix justifié dans le code par la cadence naturellement basse (5 à 15 minutes) et l'échelle modeste du champ (jusqu'à 400 × 400 m environ) : un appairage MAC

par tracker n'apporterait qu'une complexité inutile à cette échelle. Chaque récepteur filtre lui-même les messages qui lui sont destinés en comparant sa propre position à celle visée dans le paquet.

Fréquence et contenu par type de message :

Message	Émetteur	Fréquence	Contenu
Télémétrie	Tous	Périodique (cadence écriture réglée, ex. 15 min)	position, azimut, tilt, LDR moyen
Correction	Maître uniquement	Sur détection d'ombrage	position cible, un pas de tilt à appliquer
Configuration de champ	Maître uniquement	À chaque début de balayage dashboard (~1 min)	espacements X/Y, nombres de trackers X/Y
Auto-déclaration HS	Tous	Sur changement d'état uniquement	indicateur HS
Interrogation/réponse dashboard	Maître (interroge) / tracker de la rangée (répond)	Balayage complet toutes les 60 s, fenêtre de réponse 500 ms par rangée	18 champs de télémétrie étendue
Commande relayée	Maître (relais MQTT→Esclave)	À la demande	payload JSON brut + autorisation déjà vérifiée
Badges maintenance	Maître	Diffusion nocturne (23h locale) + à chaque mise à jour	liste des UID NFC autorisés

Gestion des erreurs et de la robustesse :

- Aucun accusé de réception — fonctionnement entièrement « best-effort », sans retransmission garantie (pas d'ARQ ESP-NOW).
- Un pair silencieux depuis plus de 45 minutes (3× la cadence par défaut) est exclu du calcul du cycle courant sans être supprimé de la table.
- Si le Maître reste injoignable plus de 30 minutes, l'Esclave passe en repli local : il rejoue le point le plus proche de l'heure courante dans sa propre courbe enregistrée, sans interpolation.
- Un paquet corrompu ou de taille inconnue est ignoré silencieusement.
- Le callback radio ne fait jamais de traitement lourd : toute action non triviale (écriture EEPROM, application de commande) est reportée au tick de boucle suivant.

Point non vérifié signalé dans le code. La signature exacte du callback de réception ESP-NOW a changé entre les versions du cœur Arduino-ESP32 antérieures et postérieures à la 2.0. Le firmware actuel cible une version ≥ 2.0 ; en cas d'échec de compilation sur ce point précis, le code indique la correction à apporter. Ceci n'a jamais pu être testé sur une version antérieure faute de matériel disponible en session. De même, la fenêtre d'attente de 500 ms pour les réponses dashboard est une valeur prudente issue de la documentation ESP-NOW (latences documentées entre 10 et 500 ms), **non vérifiée sur le matériel réel** — à ajuster sur site si des rangées reviennent systématiquement incomplètes.

1.5 Algorithme de détection d'ombre et de correction du tilt

Point important à connaître avant toute mise en service. L'algorithme réellement implémenté **n'effectue aucun calcul géométrique ou trigonométrique** de longueur d'ombre portée. Un commentaire du code, à côté du réglage « Hauteur du tracker », le présente comme « nécessaire au calcul géométrique de l'ombre portée » — mais aucune fonction du firmware ne lit effectivement `BacktrackHeight` pour un tel calcul (vérifié par recherche exhaustive dans tous les fichiers). Ce réglage est saisi, stocké, validé par bornes, mais **actuellement sans effet fonctionnel** sur la détection d'ombrage. `BacktrackSpacingX`/`BacktrackSpacingY` sont bien diffusés du Maître vers les Esclaves, mais eux non plus ne sont utilisés dans aucun calcul de distance ou d'angle solaire.

L'algorithme réel est une simple comparaison de seuil entre les lectures LDR de deux trackers voisins immédiats, exécutée uniquement par le Maître :

1. **Détermination du sens de balayage du soleil**, à partir de l'azimut courant (Est vs Ouest), affinée par la tendance des deux derniers points de la courbe du jour si disponibles.
2. **Pour chaque rangée indépendante**, chaque tracker regarde son voisin immédiatement en amont (côté soleil).
3. **Comparaison des LDR bruts** des deux trackers :

```
int deltaLdr = BacktrackPeerTable[idxUpstream].ldrRaw -
BacktrackPeerTable[idxSelf].ldrRaw;
if (deltaLdr < BACKTRACK_SHADOW_LDR_DELTA) continue;   // pas d'ombrage detecte
```

`BACKTRACK_SHADOW_LDR_DELTA = 150` (unités ADC brutes) est le seuil au-delà duquel le voisin amont est considéré comme projetant une ombre significative. **Ce seuil est explicitement documenté dans le code comme non vérifié sur matériel réel**, à calibrer sur site à la première mise en service (comparer deux LDR voisins par temps dégagé, lors d'un ombrage progressif au lever ou au coucher).

4. **Envoi de la correction** : seul le tracker amont (celui qui projette l'ombre) reçoit une impulsion de correction, d'amplitude égale au réglage « Pas correction tilt » (2 à 10°, par pas de 2). Ce n'est jamais une correction absolue en un coup : c'est une boucle fermée, un seul pas par cycle, réévaluée au cycle suivant.

Application côté tracker corrigé, plafonnée à 10° de correction cumulée par jour civil (remise à zéro chaque nuit) :

```
DriveTiltFromDiff(-step, 1);   // réduction de tilt (aplatir davantage)
```

> Le signe de cette correction (réduction plutôt qu'augmentation) est explicitement marqué dans le code comme **non vérifié sur le matériel réel** — à valider (et éventuellement inverser) dès le premier essai en champ.

Exclusion d'un tracker (« HS ») : manuelle, via une commande MQTT admin (`set_hs`, voir Partie 2) traitée uniquement par le Maître ; ou automatique, un tracker en mode dégradé sans donnée fiable se déclare lui-même HS et le diffuse en ESP-NOW. Un tracker marqué HS est totalement exclu du calcul, qu'il soit la position observée ou la position voisine amont.

1.6 Écran grille « Position dans le champ »

Écran tactile dédié (plutôt qu'un simple champ numérique), pour permettre de visualiser directement la place du tracker dans le champ plutôt que de saisir des coordonnées à l'aveugle. Une grille correspondant au nombre de trackers X × Y est dessinée, chaque cellule représentant un tracker ; la cellule correspondant à la position actuelle est mise en surbrillance.

La taille des cellules est calculée dynamiquement selon la densité de la grille (plafonnée à 36 px, plancher à 6 px) — **sans pagination ni défilement** : au-delà d'une grille très dense, les cellules rétrécissent jusqu'au minimum plutôt que de déborder ou de nécessiter un défilement, un choix de conception qui privilégie la robustesse (« jamais planter ») au confort visuel dans les cas extrêmes.

Double voie de saisie, chacune suffisante seule : tactile (un tap direct sur une cellule déplace immédiatement la sélection) ou encodeur rotatif (parcourt la grille ligne par ligne). Un bouton « Valider » ou le bouton de l'encodeur referme l'écran ; la position n'est réellement persistée qu'à la sortie complète du sous-menu Backtracking, pas à chaque déplacement de cellule. Si le nombre de trackers réglé est ensuite réduit, la position enregistrée est automatiquement re-bornée pour ne jamais pointer hors grille.

1.7 Mécanisme de licence (option payante)

Le réglage « Activation » du Backtracking ne peut passer sur On que si l'état interne `BacktrackUnlocked` est vrai ; toute tentative sans déverrouillage est immédiatement annulée avec un message « Backtracking verrouillé — Option payante ».

Principe cryptographique. L'identifiant matériel du tracker est lu directement dans le registre eFuse de la puce (`ESP.getEfuseMac()`, disponible dès le démarrage, indépendamment de l'initialisation Wi-Fi), formaté en six octets hexadécimaux séparés par « : ». Le code de déverrouillage attendu est calculé — **jamais stocké, toujours recalculé à la volée à chaque vérification** — ainsi :

```
code = HMAC-SHA256(secret_partage, identifiant_materiel)
       → 4 premiers octets → 8 caracteres hexadecimaux majuscules
```

Le même secret partagé sert aussi de base à une clé AES-128 (dérivée par simple hachage SHA-256 du secret, pour éviter de gérer une seconde constante secrète). Elle chiffre un bloc de 16 octets contenant un magic d'intégrité, l'état déverrouillé (0/1), une réserve à zéro, et un CRC32 de contrôle — chiffré en AES-128 ECB (mode simple bloc, justifié par le fait que le contenu protégé tient exactement dans un seul bloc de 16 octets). Au déchiffrement, le magic et le CRC sont revérifiés avant d'accepter la valeur : aucune donnée partielle ou corrompue n'est jamais prise pour valide.

Écran de déverrouillage : affiche l'identifiant matériel (à transmettre au fournisseur) et l'état actuel (déverrouillé / verrouillé). Deux sorties possibles : ouvrir la saisie du code à 8 caractères, ou un bouton « Retour » tactile qui sort sans rien saisir. Le code saisi est comparé au code recalculé ; en cas de correspondance, l'état déverrouillé est immédiatement persisté à la fois sur la carte SD et, en cascade, dans la copie de secours EEPROM.

Persistance au redémarrage : le bloc EEPROM principal ne contient pas l'état de licence. Au boot, la copie de secours EEPROM restaure d'abord une valeur par défaut/de secours, puis la carte SD (considérée prioritaire) écrase cette valeur si elle est elle-même valide. **Il n'y a donc aucune revérification cryptographique en ligne au démarrage** : une fois acquis, l'état déverrouillé est simplement relu depuis son stockage chiffré local à chaque redémarrage, sans nouvelle saisie de code.

> **Limite de sécurité honnêtement documentée dans le code.** Cette protection est un « verrou logiciel » standard, qui dissuade un contournement casuel — ce n'est **pas** une protection cryptographique inviolable contre un attaquant déterminé ayant un accès physique et pratiquant la rétro-ingénierie d'un dump de la mémoire flash : quiconque lit le firmware compilé peut en principe retrouver la chaîne secrète qui y est embarquée. Le secret utilisé pour le Backtracking est **totalement indépendant** de celui utilisé pour l'authenticité des mises à jour OTA locales (deux enjeux, deux clés séparées, par conception délibérée) — voir le Manuel Utilisateur pour la mise à jour OTA.

1.8 Indicateur d'état sur l'écran Mode Run

Une fonction d'affichage dédiée existe et est fonctionnellement complète dans le firmware : elle peut afficher « BRC -- » (désactivé), « BRC M OK » / « BRC S OK » (actif, rôle Maître/Esclave, radio saine), ou « BRC M/S ERR » en clignotement rouge (actif mais communication perdue). **Cependant, son appel a été explicitement retiré de l'écran Mode Run** dans le firmware actuel — la fonction est laissée intacte mais « dormante », non supprimée au cas où elle serait réactivée plus tard.

Conséquence pratique pour l'installateur : à ce jour, l'écran Mode Run n'affiche **aucun** indicateur visuel de l'état du Backtracking. Le seul moyen de connaître cet état est le sous-menu Admin > Backtracking (rôle, activation) ou, côté Maître, le tableau de bord/MQTT décrit en Partie 2.

1.9 Synthèse des points de vigilance — Backtracking

- Le réglage « Hauteur du tracker » n'a actuellement **aucun effet** sur le calcul d'ombrage malgré son commentaire dans le code ; l'algorithme réel est une simple comparaison de seuil LDR entre voisins.
- Le seuil de détection (150, unités ADC brutes) et le sens de la correction de tilt sont explicitement marqués comme **non vérifiés sur matériel réel** — à calibrer/valider impérativement à la première mise en service sur site.
- La convention d'axes (position X = sens de balayage du soleil, position Y = rangées indépendantes) est une hypothèse de conception à vérifier/adapter selon le montage réel du champ.
- La table des trackers connus n'est jamais persistée sur le Maître : un redémarrage efface toute la connaissance de la flotte, y compris les exclusions HS manuelles.
- Aucun accusé de réception ESP-NOW : la robustesse repose entièrement sur des délais de silence (pair distant à 45 min, Maître silencieux à 30 min), pas sur une confirmation de livraison.
- L'indicateur visuel d'état Backtracking sur l'écran Mode Run est actuellement désactivé (fonction présente mais non affichée).
- La licence est un verrou logiciel dissuasif, pas une protection inviolable contre un accès physique déterminé au firmware compilé.

Partie 2 — MQTT : documentation du protocole

2.0 Bibliothèque et architecture des connexions

Le firmware utilise la bibliothèque **PubSubClient**, sur une couche de transport `WiFiClientSecure` — c'est l'unique bibliothèque tierce du projet pour la couche MQTT/TLS (réécrire cette couche à la main a été jugé hors de portée raisonnable). Toute la gestion réseau, y compris MQTT, est portée par une tâche dédiée tournant sur le second cœur du processeur (« core1 »), à une cadence de boucle de 200 ms.

Le firmware gère **jusqu'à quatre connexions distinctes**, mais jamais plus d'une connexion MQTT « maison » active à la fois :

Rôle	Usage
Broker « Client »	Broker fourni par le client final (ex. HiveMQ Cloud) — priorité absolue s'il est configuré
Broker « Propriétaire 1 »	Broker de secours fourni par le fabricant, utilisé seulement si le broker Client est vide
Broker « Propriétaire 2 »	Second broker de secours, bascule automatique avec le n°1
Tuya Cloud (TuyaLink)	Connexion entièrement séparée — voir §2.9

2.1 Broker « Client »

Champs : hôte, utilisateur, mot de passe, topic (valeur par défaut du topic : `tracker/status`). Saisissables par deux chemins équivalents, qui écrivent les mêmes variables internes : l'écran local (Admin > Sans-fil > Menu MQTT) ou le portail Wi-Fi captif du téléphone (SSID `TRACKER-SETUP`).

Le champ « hôte » accepte soit un simple nom d'hôte, soit une URL complète copiée depuis la console d'un fournisseur cloud (ex. `xxxx.s1.eu.hivemq.cloud:8884/mqtt`) : le firmware retire automatiquement le schéma, le chemin et le port pour n'en garder que le nom d'hôte. **Le port est figé à 8883 (TLS)** pour ce broker, quelle que soit une éventuelle ancienne valeur enregistrée.

Identifiant client MQTT : c'est simplement le nom d'utilisateur configuré (ou « tracker » par défaut) — aucun identifiant matériel unique (adresse MAC, numéro de série) n'est utilisé comme Client ID.

2.2 Brokers « Propriétaire 1 » et « Propriétaire 2 »

Saisis **uniquement à l'écran local** (pas via le portail téléphone). Utilisés seulement si le champ hôte du broker Client est vide. Le broker « Propriétaire 2 » n'a pas encore de champ Topic dédié à ce jour et retombe provisoirement sur le topic du broker Client — limite de périmètre explicitement assumée dans le code (« mise en œuvre à un seul broker prioritaire, la redondance à deux reste partielle »).

2.3 Chiffrement TLS

Pour tous les brokers « maison » comme pour Tuya, le firmware établit une connexion TLS **sans vérification du certificat serveur** (mode dit « insecure » de la bibliothèque). La connexion est donc chiffrée, mais un attaquant en position d'interception sur le réseau pourrait en principe présenter un faux certificat sans que le tracker ne le détecte. C'est un compromis délibéré (éviter d'embarquer et de maintenir la chaîne de certificats exacte du fournisseur de broker), documenté comme tel dans le code — pas un oubli, mais une limite de sécurité réelle à connaître.

2.4 Topics MQTT utilisés

Le topic configuré (ex. `tracker/status`) sert de racine à tous les topics « maison » :

Topic	Sens	Contenu
`<topic>`	Publication (tracker → dashboard)	Message « status » (toutes les 3 s) et message étendu « ex » (toutes les 30 s), sur le même topic
`<topic>/cmd`	Abonnement (dashboard → tracker)	Commandes JSON `{"cmd":"..."}`, voir §2.5
`<topic>/+/cmd`	Abonnement (wildcard)	Capte tous les `<topic>/slave_<X>_<Y>/cmd` sans connaître à l'avance la taille de la flotte
`<topic>/slave_<X>_<Y>`	Publication	Un message par tracker connu de la flotte Backtracking (y compris le Maître sous son propre alias), pour le

Topic	Sens	Contenu
		tableau de bord « flotte »
`<topic>/slave_<X>_<Y>/cmd`	Abonnement (via wildcard)	Commande adressée à un tracker Esclave précis (relayée en ESP-NOW) ou au Maître lui-même

2.5 Commandes reçues à distance

Point d'entrée unique dans le firmware, avec un analyseur JSON maison minimal (pas de bibliothèque JSON générique) ; tout message sans champ `"cmd"` reconnu est ignoré silencieusement. Deux niveaux d'autorisation existent : aucune garde pour certaines commandes, ou vérification qu'une session Admin/Root est active localement **ou** qu'un jeton distant valide a été fourni (voir §2.8).

Commande	Garde requise	Effet
`{"cmd":"auto"}`	Aucune (bloquée si calibrage jamais fait)	Bascule en suivi Automatique
`{"cmd":"manu"}`	Aucune	Bascule en Manuel, coupe les moteurs
`{"cmd":"axis"}`	Aucune	Armer/confirmer : bascule 1 axe / 2 axes
`{"cmd":"repli"}`	Oui	Bascule la protection auto-repli vent
`{"cmd":"maint_code","code":"..."}`	Oui	Pousse un code de bypass maintenance tilt
`{"cmd":"maint_badges","uids":"..."}`	Oui	Met à jour le cache de badges NFC autorisés
`{"cmd":"estop"}`	**Aucune, volontairement**	Coupe immédiatement les moteurs — un seul envoi suffit, pas de cycle armer/confirmer
`{"cmd":"reset"}`	Oui	Redémarre l'appareil
`{"cmd":"force_secours"}`	Oui	Force le rejeu de la courbe de secours
`{"cmd":"force_repli"}`	Oui	Force le tilt en position de repli
`{"cmd":"force_clear"}`	Oui	Lève les deux forçages ci-dessus
`{"cmd":"set_hs","x":N,"y":N,"hs":1|0}`	Oui + rôle Maître uniquement	Marque/lève HS un tracker de la flotte
`{"cmd":"set_remote_token","token":"..."}`	Session Admin locale uniquement (jamais le jeton lui-même)	Change le jeton distant
`{"cmd":"jog","dir":"east|west|up|down","state":"start|stop"}`	Éligibilité Manuel + pas d'alarme vent + pas d'estop verrouillé	Démarre/arrête un mouvement manuel

Commande	Garde requise	Effet
`{"cmd":"raz_az"}` / `{"cmd":"raz_tilt"}`	Oui	Armer/confirmer : calibration du zéro azimut/tilt
`{"cmd":"nuit"}`	Aucune	Armer/confirmer : recalibre le seuil de nuit
`{"cmd":"ldr_offset","which":"E|O|H|B","delta":N}`	Aucune	Ajuste l'offset d'un capteur LDR
`{"cmd":"auto_src","src":"LDR|SEC|CRB"}`	Aucune (sauf alarme vent/estop verrouillé)	Choisit la source du suivi Auto

> Piège documenté explicitement dans le code : le champ `"hs"` de `set_hs` doit être `0` ou `1` numérique, pas un booléen JSON `true`/`false` — l'analyseur maison le convertirait en `0` dans les deux cas.

2.6 Comparaison avec le tableau de bord web

Le tableau de bord web est un **vrai client MQTT exécuté dans le navigateur** (bibliothèque JavaScript dédiée), connecté en WebSocket sécurisé directement au même broker que le tracker — ce n'est ni un mécanisme propriétaire ni un serveur intermédiaire. Chaque bouton correspond, un pour un, à une commande ci-dessus : AUTO, MANU, SECU (→ `repli`), AXE, RAZ AZ, RAZ TILT, NUIT (cycle armer/confirmer reproduit côté navigateur).

Deux différences à connaître : **EXIT n'envoie strictement aucune commande MQTT** — c'est un retour visuel purement local au navigateur — et les boutons **REC et LEVER de l'écran physique ont été retirés du tableau de bord web**, car aucune commande `"rec"` ni `"lever"` n'existe dans le protocole MQTT réel.

2.7 Données publiées automatiquement

Trois messages distincts, tous construits côté tracking (« core0 ») puis transmis à la tâche réseau (« core1 ») qui les publie réellement :

Message « status » — même topic que la racine, toutes les 3 secondes :

```
{"az":<azimut>,"ti":<tilt>,"md":"<mode>","sf":"<securite>","nt":<nuit>,
 "ax":"<axe>","es":<estop>,
 "lE":<LDR Est>,"lO":<LDR Ouest>,"lH":<LDR Haut>,"lB":<LDR Bas>,
 "wd":<vent>,"wM":<seuil vent>,"bt":<batterie>,"fb":<fallback>,"rp":<repli>,
 "aM":<moteur az actif>,"tM":<moteur tilt actif>,"wa":<secondes alarme
vent>,"ha":<alarme grele>}
```

Message étendu « ex » — même topic, toutes les 30 secondes : calibrage azimut/tilt, seuil nuit, offsets LDR, codes météo, source de suivi active, alarme HS, prévisions, heure locale, état I2C, indicateurs capteurs, état de la fenêtre de repli, etc.

Message flotte — topic `<topic>/slave_<X>_<Y>`, un par tracker connu, publié lors de chaque balayage complet (cadence réelle environ 1 minute), champs similaires au message « status » plus l'état HS et le bypass maintenance.

2.8 Sécurité

- **Authentification broker** : identifiants MQTT classiques (utilisateur/mot de passe) au niveau de la connexion, jamais codés en dur dans le firmware ; un utilisateur vide est accepté (broker anonyme).
- **Chiffrement** : TLS actif partout, mais sans vérification de certificat (voir §2.3).
- **Restriction par topic (ACL)** : le firmware ne configure ni ne vérifie aucune ACL côté broker — cette responsabilité reste entièrement à la charge de l'installateur (limiter chaque identifiant client à son propre topic, côté broker).
- **Jeton distant (« Admin »)** : mécanisme MQTT réel et applicatif — une chaîne de caractères persistée en mémoire, vide par défaut (mécanisme désactivé tant qu'elle l'est). Vérifiée une seule fois, en comparant un champ `"token"` du message JSON reçu à la valeur enregistrée ; le résultat de cette vérification est propagé sans jamais être recalculé plus loin dans la chaîne de traitement. Un jeton valide autorise exactement les mêmes commandes qu'une session Admin locale, mais ne peut jamais servir à changer le jeton lui-même (seule une vraie session Admin locale le peut) — protection délibérée contre un jeton compromis qui se renouvellerait indéfiniment. Le lien/QR de configuration généré pour le dashboard transmet l'hôte, l'utilisateur, le mot de passe et le topic, **mais jamais le jeton**, qui doit être communiqué séparément.
- **Arrêt d'urgence** : le **déverrouillage** de l'arrêt d'urgence n'est jamais accessible à distance (MQTT ou web) — uniquement en local, depuis le menu Admin. Le **déclenchement**, en revanche, reste toujours autorisé à distance, sans aucune garde, y compris pendant une alarme vent — cohérent avec le principe que l'arrêt d'urgence doit rester la commande la plus disponible du système.

2.9 Lien avec l'intégration Tuya

L'intégration Tuya Cloud utilise un **client et un broker MQTT entièrement séparés** de la connexion « maison » décrite ci-dessus (serveur dédié, identifiants Tuya propres, schéma de signature spécifique à la plateforme Tuya). Les commandes Tuya reçues sont traduites en interne vers les mêmes commandes internes que celles du tableau de bord web (`{"cmd":"auto"}`, etc.) et passent par le même point d'entrée de traitement. **Cette connexion Tuya Cloud n'a jamais été testée sur matériel réel** dans le cadre de ce développement — son adresse serveur et son format de signature ont été vérifiés uniquement par rapport à la documentation officielle Tuya, à valider impérativement lors de la première mise en service Tuya sur site.

2.10 Reconnexion et robustesse

- Tentative de reconnexion limitée à une fois toutes les 5 secondes (identique pour le broker maison et pour Tuya).
- La boucle de maintenance de connexion (ping, réception des messages) est appelée à chaque tour de la tâche réseau, soit toutes les 200 ms — une cadence plus lente ferait tomber la connexion en pratique.
- Bascule automatique entre les deux brokers « Propriétaire » (uniquement si le broker Client est vide) : bascule immédiate sur coupure détectée, ou nouvelle tentative sur le premier après 10 minutes sans connexion sur le second.
- **Aucun tampon hors ligne** : une publication échoue silencieusement si le broker est injoignable — les données ne sont jamais mises en file d'attente, le prochain instantané (3 s / 30 s) écrase simplement le précédent. Une commande reçue pendant une coupure de connexion est, par nature, perdue.
- La taille de tampon MQTT a été explicitement portée à 512 octets après un bug réel : le message étendu « ex » dépassait la limite par défaut de 256 octets, provoquant un échec silencieux de publication (aucune donnée météo/heure reçue côté dashboard, sans aucune erreur visible) — corrigé, mais à garder en tête pour tout ajout futur de champs à ces messages.

2.11 Synthèse des points de vigilance — MQTT

- Aucune vérification de certificat TLS (mode « insecure ») sur l'ensemble des connexions MQTT du firmware — chiffré, mais pas authentifié côté serveur.
- Aucune ACL gérée par le firmware : la segmentation des topics par tracker reste entièrement à la charge de l'installateur, côté broker.

- Le broker de secours « Propriétaire 2 » ne dispose pas encore de son propre topic et retombe sur celui du broker Client — redondance actuellement partielle.
- L'intégration Tuya Cloud n'a jamais été validée sur matériel réel.
- Aucun tampon hors ligne : toute commande ou donnée transmise pendant une coupure réseau est perdue, sans rejeu après reconnexion.
- Le texte d'état MQTT affiché à l'écran (« connecté », « erreur », etc.) est signalé dans le code comme non pleinement validé sur matériel réel.