

Pertes d'ordres d'aiguillage en exploitation automatique Rocrail

Rapport d'investigation

Réseau numérique Märklin 3 rails, échelle HO

Version 1.0 — document de travail

Auteur : Jerome UK Forum 3 rails & Opus Claude

Période couverte : août – septembre 2026

État : chaîne amont résolue ; décodeurs de servos en suspens

1. Énoncé du problème

En exploitation automatique sous Rocrail, des ordres d'aiguillage étaient perdus. Le TCO de Rocrail affichait l'aiguille comme manœuvrée, et son journal enregistrait bien l'ordre comme émis, mais l'aiguille ne bougeait pas physiquement.

Caractéristiques du défaut tel que constaté à l'origine :

- **Intermittent et imprévisible** — une défaillance pouvait survenir après 30 secondes comme après 10 minutes de circulation.
- **Non spécifique à une aiguille** — des adresses différentes échouaient selon les occasions.
- **Corrélé à la charge** — bien plus fréquent en exploitation automatique, où le trafic d'ordres est dense, qu'en commande manuelle.
- **Lourd de conséquences** — une aiguille restée en mauvaise position envoyait les trains sur des itinéraires non prévus, provoquant des détections de train fantôme, des arrêts d'urgence et, à plusieurs reprises, des quasi-collisions imposant une coupure manuelle de l'alimentation.

Conséquence pratique : l'exploitation automatique sans surveillance n'était pas envisageable. Rocrail verrouille un itinéraire une fois celui-ci accordé ; l'opérateur ne peut donc pas corriger manuellement une aiguille appartenant à un itinéraire verrouillé. Un seul ordre manqué ne peut alors se résoudre qu'en arrêtant le réseau.

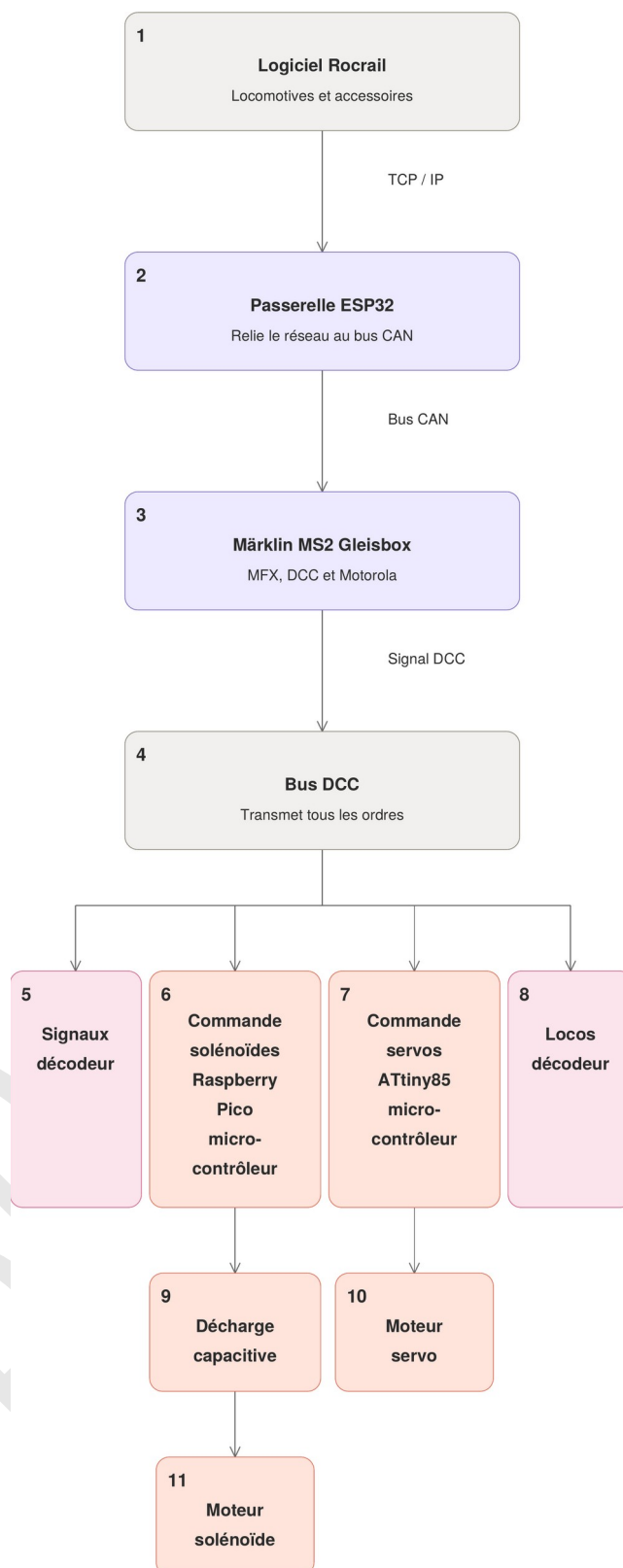
1.1 Système étudié

La chaîne de commande des accessoires et des locomotives est détaillée ci-dessous. Chaque étage numéroté correspond au schéma d'architecture de la section 1.2.

N°	Étage	Rôle
1	Logiciel Rocrail (PC)	Automatisation ; émet les ordres pour locomotives et accessoires
2	Passerelle ESP32	Relie le réseau TCP/IP (WiFi ou Ethernet) au bus CAN Märklin
3	Märklin MS2 Gleisbox	Convertit les ordres CAN en signaux de voie (MFX, DCC, Motorola)
4	Bus DCC	Environ 4 m, filtre RC snubber à l'extrémité ; transmet tous les ordres
5	Décodeurs de signaux	Signalisation latérale
6	Décodeurs d'aiguilles à solénoïde	Raspberry Pico, 8 adresses par carte
7	Décodeurs d'aiguilles à servo	ATTiny85, une adresse par carte
8	Décodeurs de locomotives	Embarqués
9	Décharge capacitive	Une carte et un condensateur par aiguille à solénoïde
10	Moteur servo	Actionnement physique des aiguilles à servo
11	Moteur solénoïde	Actionnement physique des aiguilles à solénoïde

Parc d'aiguilles : 11 aiguilles à solénoïde (adresses 150–160, deux cartes Pico, chaque aiguille disposant de sa propre carte de décharge capacitive) et 8 aiguilles à servo (adresses 167–174, huit cartes ATTiny85 indépendantes).

1.2 Architecture



Chaîne de commande, de Rocrail jusqu'aux moteurs d'aiguille. Les étages 1 à 4 sont communs à tous les types d'ordres ; la chaîne se divise au niveau du bus DCC.

2. Démarche d'investigation

La difficulté centrale au départ était qu'aucun étage de la chaîne ne pouvait être ni confirmé ni mis hors de cause. Le journal de Rocrail consigne ce qu'il a l'intention d'émettre ; rien n'enregistre ce qui parvenait réellement aux rails. Toute hypothèse était donc invérifiable.

L'investigation a été réorganisée autour de l'obtention d'une **mesure de référence à chaque maillon**.

2.1 Analyseur (sniffer) DCC

Un moniteur de bus DCC autonome a été construit sur Raspberry Pico, en réutilisant le dessin de circuit imprimé des décodeurs existants (optocoupleur 6N137 pour l'isolation galvanique, entrée sur GP2).

Point déterminant : l'analyseur mesure la durée des bits DCC au moyen du **matériel PIO du RP2040** et non d'interruptions logicielles. Les impulsions sont mesurées à la microseconde par un périphérique dédié, que rien de ce que fait le processeur ne peut retarder. Ce choix s'est avéré décisif — voir section 4.3.

L'analyseur reconstitue les trames complètes, vérifie la somme de contrôle XOR, décode les trames d'accessoires et émet une ligne horodatée par trame. Le décodage d'adresse a été calibré expérimentalement : une aiguille d'adresse connue a été commandée manuellement et l'adresse décodée par l'analyseur vérifiée comme identique.

2.2 Outils de journalisation et de réconciliation

Trois outils ont été développés :

- **dcc_logger.py** — capture un flux série dans un fichier horodaté, en préfixant chaque ligne de l'heure du PC afin de pouvoir corréler des journaux indépendants. Une instance par appareil (analyseur, passerelle).
- **dcc_reconcile.py** — analyse le fichier de trace de Rocrail et le journal de l'analyseur, et signale tout ordre d'aiguillage émis par Rocrail n'ayant produit aucune trame correspondante sur les rails. Il indique aussi le bit de direction observé pour chaque ordre apparié et signale toute incohérence.
- **Micrologiciel instrumenté de la passerelle** — compteurs de pertes à chaque endroit où une trame peut être abandonnée, ainsi que des lignes [ACC-REQ] et [ACC-RESP] marquant l'arrivée de chaque ordre d'accessoire depuis Rocrail et chaque acquittement du Gleisbox.

Ensemble, ces outils donnent une visibilité complète : l'intention de Rocrail, la réception et la retransmission par la passerelle, l'acquiescement du Gleisbox, et le signal physique sur les rails.

2.3 Deux points de méthode à retenir

L'observation visuelle sous-estime le taux de défaillance. Au début de l'investigation, tous les trains circulaient dans le même sens ; une aiguille défaillante prise en talon n'avait donc aucune conséquence visible. La réconciliation a révélé des pertes silencieuses environ deux fois plus nombreuses que les pertes observées. L'inspection visuelle est également ambiguë lorsque l'aiguille était déjà dans la position commandée : un ordre perdu et un ordre exécuté sont alors indiscernables.

La réconciliation doit exclure les ordres répétés à état identique. Rocrail consigne une ligne « set switch » même lorsqu'il commande un état que l'aiguille occupe déjà, mais aucune nouvelle trame DCC n'est alors générée. Une première version de l'outil comptabilisait ces cas comme des pertes et annonçait un taux d'échec de 20 sur 36, en grande partie artificiel. Seuls les véritables changements d'état sont évalués.

3. Étages examinés et mis hors de cause

Étage	Méthode	Résultat
Bus DCC, câblage, filtre RC	Analyseur raccordé successivement en sortie de Gleisbox et à l'extrémité opposée, près des décodeurs	Résultats identiques — les pertes se produisaient avant même que le signal n'atteigne les rails. Bus hors de cause.
Transport réseau	Passerelle testée en Ethernet puis en WiFi, micrologiciel et réglages identiques	Aucune différence mesurable ; les deux sont propres. Le WiFi est conservé pour une installation plus soignée.
Fonctionnement interne de la passerelle	Compteurs de pertes aux trois points de file et d'émission	Zéro perte sur l'ensemble des sessions. Passerelle hors de cause comme source de perte.
Recharge des condensateurs (solénoïdes)	Calcul ($1\text{ k}\Omega \times 1000\text{ }\mu\text{F} \approx 3\text{ s}$ pour 95 %) et essais manuels	Réel, et corrigé — voir 4.2. N'explique pas les défaillances en mode automatique.
Messages d'enclenchement Rocrail	Analyse des journaux	Les mentions « switches check failed » et « is locked by » relèvent du fonctionnement normal des enclenchements, pas de défaillances. Fausse piste.

4. Défauts identifiés et correctifs déployés

Six défauts distincts ont été identifiés. Ils sont présentés dans l'ordre de leur découverte.

4.1 Décodeur solénoïde : case de commande unique (corrigé)

Défaut. Le décodeur Pico ne conservait qu'une seule commande en attente. Si un second ordre destiné à une autre sortie de la même carte arrivait pendant qu'une impulsion était en cours, il écrasait le premier, qui était alors abandonné silencieusement.

Correctif. Remplacement par une file circulaire (profondeur 8, une case par sortie). Les ordres sont traités dans l'ordre ; aucun n'est perdu.

4.2 Décodeur solénoïde : durée d'impulsion excessive (corrigé)

Défaut. SOLENOID_PULSE_MS valait 500 ms. Chaque condensateur se recharge à travers une résistance de 1 kΩ avec une constante de temps d'environ 1 s, soit environ 3 s pour atteindre 95 %. Une impulsion de 500 ms consommait la charge et retardait le début de la recharge.

Correctif. Réduction à 50 ms.

Résultat. La commande manuelle est devenue parfaitement fiable, y compris pour des ordres rapprochés sur la même aiguille ou sur des aiguilles différentes. L'exploitation automatique est restée inchangée, ce qui indiquait justement une seconde cause indépendante.

4.3 Décodeur solénoïde : réception DCC par interruptions (corrigé)

Défaut. Les décodeurs utilisaient la bibliothèque DCC_Decoder, qui mesure les bits DCC au moyen d'interruptions processeur. Sous trafic dense, le processeur peut être occupé au moment où survient un front critique, ce qui décale la mesure hors de sa fenêtre valide : la trame est alors mal interprétée ou perdue.

Preuve. La démonstration reposait sur une expérience naturelle déjà présente dans l'installation : l'analyseur, lisant le même bus au même instant en PIO, n'a jamais manqué une trame, là où les décodeurs à interruptions en manquaient. Les sessions ont montré à plusieurs reprises un ordre acquitté par le Gleisbox, confirmé valide sur les rails par l'analyseur, et non exécuté par le décodeur.

Correctif. Réécriture de la réception à partir du programme PIO de mesure d'impulsions et de la machine à états de reconstitution de trames de l'analyseur (décodeur v0.7).

Résultat. Une comparaison contrôlée a été menée avec une carte Pico mise à jour et l'autre laissée en ancienne version : la carte mise à jour a traité une rafale dense sans défaillance tandis que l'autre échouait dans la même session. Depuis la mise à jour des deux cartes, aucune défaillance de solénoïde n'a été observée.

4.4 Saturation du Gleisbox (atténué)

Défaut. Lors de rafales d'ordres denses, le Gleisbox cesse d'acquitter et ne produit aucune sortie DCC pour les ordres concernés. Le motif est sans ambiguïté : sur une rafale capturée de huit ordres en environ 800 ms, les trois premiers ont été acquittés et transmis, les quatre suivants n'ont été ni acquittés ni transmis, et le huitième a été acquitté et transmis — le comportement d'une mémoire tampon saturée qui se vide puis redémarre.

Fondement du correctif. La spécification du protocole CAN de Märklin (document CS2, section 1.2.5) indique que les messages CAN ne sont pas transmis de manière fiable, que leur réception n'est pas confirmée, et que l'exécution n'est confirmée que par un message de confirmation — et qu'en l'absence de cet acquittement, on peut considérer que l'action n'a pas été exécutée. Ni Rocrail ni la passerelle n'exploitaient ce mécanisme.

Correctif. La passerelle v2.0.0 suit chaque ordre d'accessoire jusqu'au retour de son acquittement (appariement sur les six premiers octets de données, ce qui distingue l'activation de la désactivation) et ne le réémet que si aucun acquittement n'arrive dans les 150 ms. Il s'agit d'une reprise en boucle fermée conforme à l'esprit du protocole, et non d'une répétition aveugle — le délai laisse aussi le temps à un Gleisbox saturé de se vider avant d'ajouter de la charge.

Ajustement. La v2.0.0 autorisait 3 renvois (environ 450 ms de couverture). Les mesures ont montré des silences du Gleisbox dépassant 500 ms ; une session a perdu 8 ordres avec seulement 1 renvoi réussi sur 27. La v2.1.0 porte la limite à 8 renvois (environ 1,2 s). Lors de la session suivante, les 8 ordres nécessitant un renvoi ont tous été récupérés et aucun n'a été abandonné.

4.5 Temps de commutation d'itinéraire Rocrail (corrigé par paramétrage)

Défaut. Le paramètre « Route switch time » de Rocrail (onglet Automatic) était réglé à 150 ms. Il s'agit de l'intervalle auquel Rocrail parcourt les aiguilles d'un itinéraire, et il correspondait exactement à l'espacement mesuré des ordres dans toutes les rafales problématiques : la saturation décrite ci-dessus était donc directement provoquée par ce réglage.

Correctif. Porté à 500 ms.

Résultat. Une session de 17 minutes comportant 109 véritables changements d'état n'a enregistré aucune perte sur le bus — la première session entièrement propre de cette durée depuis le début de l'investigation.

Contrepartie. Un itinéraire de six aiguilles demande désormais environ 3 s au lieu de 0,9 s. Les trains patientent un peu plus longtemps avant de partir, mais ne circulent pas plus lentement.

4.6 Décodeur de servo : condition de course et décalage de position supposée (partiellement corrigé)

Deux défauts avérés ont été trouvés dans le micrologiciel ATtiny85.

Condition de course (v0.8). Le gestionnaire DCC s'exécutait en contexte d'interruption et écrivait directement dans la variable state, que la boucle principale écrit également. Un ordre arrivant pendant les transitions MOVING vers HOLD ou HOLD vers WAIT_COMMAND était écrasé et perdu. Corrigé en faisant positionner à l'interruption un drapeau indépendant que la machine à états consomme ; l'interruption ne touche plus à state.

Décalage de position supposée (v0.9). currentPos n'est que la croyance du logiciel sur la position du servo ; elle vaut ANGLE_OPEN à la mise sous tension quelle que soit la réalité physique et n'était jamais recalée. Lorsqu'un ordre demandait la position déjà supposée occupée, la machine à états émettait une seule impulsion puis détachait le servo quelques microsecondes plus tard : celui-ci ne bougeait jamais. Une fois croyance et réalité désynchronisées, le décodeur refusait silencieusement de se corriger.

Ce mécanisme a été identifié grâce à une observation : après une mise sous tension à froid, le premier ordre manuel ne fonctionnait jamais et une seconde tentative était toujours nécessaire — correspondance exacte avec le défaut décrit.

Corrigé en attachant le servo à chaque ordre, en le pilotant vers la cible et en maintenant la position pendant 700 ms avant de le détacher, de sorte que chaque ordre soit auto-correcteur.

Résultat. Le symptôme de démarrage à froid a disparu, confirmant le diagnostic. Les défaillances de fond n'ont pas été résolues — voir section 6.

5. Erreurs et fausses pistes

Consignées par souci d'exhaustivité, plusieurs ayant coûté un temps notable.

- **Une régression a été introduite dans le micrologiciel de servo v0.7.** Une « optimisation » évitait de piloter le servo lorsque `currentPos` égalait `targetPos`. Cette croyance pouvant être fausse, le décodeur refusait définitivement de se corriger, et les deux cartes mises à jour ont échoué de façon répétée. Revert effectué. Le risque sous-jacent a ensuite été correctement diagnostiqué et traité en v0.9.
- **Trois modifications ont été groupées dans le micrologiciel de servo v0.7** (correction de la course, course plus rapide, optimisation d'attache). Le résultat étant plus mauvais, la cause était ambiguë. La v0.8 est revenue à une seule modification. Une variable à la fois.
- **L'affirmation selon laquelle Rocrail pourrait être configuré pour réémettre un ordre d'aiguillage n'a pas pu être étayée** et a été retirée. Aucun réglage de ce type n'a été trouvé.
- **Du temps a été consacré aux niveaux de trace de Rocrail** en supposant que les lignes « set switch » étaient filtrées. Elles étaient absentes simplement parce que les fragments de trace examinés couvraient des périodes sans aucun ordre d'aiguillage.
- **Le décodeur Pico v0.6 n'a d'abord pas fonctionné du tout**, sans réaction à aucun ordre. Une version de diagnostic a prouvé que la réception était parfaite ; le même code reflashé en v0.7 a fonctionné. La cause était très probablement un flashage incomplet ou périmé.

6. Constat actuel

6.1 État par étage

Étage	État
Rocrail vers passerelle (WiFi ou Ethernet)	Propre — aucune perte observée sur l'un ou l'autre transport
Fonctionnement interne de la passerelle	Propre — zéro perte sur toutes les sessions
Passerelle vers Gleisbox	Atténué — la reprise sur acquittement rattrape les décrochages du Gleisbox
Gleisbox vers bus DCC	Propre avec un temps de commutation d'itinéraire de 500 ms
Décodeurs solénoïdes (11 aiguilles)	Résolu — aucune défaillance depuis le passage en PIO
Décodeurs de servos (8 aiguilles)	EN SUSPENS — seule source de défaillance restante

6.2 Données de sessions représentatives

Session	Configuration	Ordres évalués	Perdus sur le bus	Défaillances observées
Début	Micrologiciels d'origine	15	3 (20 %)	1 visible, 2 silencieuses
Début	Micrologiciels d'origine	7	3 (43 %)	1 visible, 2 silencieuses
Début	Prise amont au Gleisbox	13	2 (15 %)	Position sur le bus indifférente
Milieu	Passerelle v2.0.0, 3 renvois	79	1	8 abandonnés par la passerelle
Fin	Passerelle v2.1.0, 8 renvois, 500 ms	109	0	1 servo (sw26)
Fin	Passerelle WiFi v2.1.0, 500 ms	43	0	1 servo (sw28)
Fin	Passerelle WiFi v2.1.0, 500 ms	19	0	1 servo (sw26)

6.3 Le problème résiduel

Toutes les défaillances restantes se limitent aux huit aiguilles à servo (adresses 167–174), chacune pilotée par son propre décodeur ATtiny85. Les onze aiguilles à solénoïde utilisent le même protocole DCC sur le même bus et n'ont plus connu la moindre défaillance depuis la réécriture de leurs décodeurs en réception PIO. Depuis le déploiement des correctifs de la passerelle et du temps de commutation d'itinéraire, aucune défaillance n'est imputable à un autre étage de la chaîne.

Toutes les défaillances désormais observées présentent la même signature : **l'ordre est acquitté par le Gleisbox, une trame DCC valide portant le bon bit de direction est confirmée sur les rails par l'analyseur, et le servo ne bouge pas.**

L'aiguille sw26 (adresse 167) a désormais échoué de cette manière lors de quatre sessions consécutives, dont deux depuis le micrologiciel v0.9. Ses ordres ont été confirmés propres sur le bus à chaque fois.

La gravité s'exprime le mieux en durée de session. Les sessions se terminent actuellement au bout de 3 à 6 minutes environ, lorsqu'une défaillance de servo provoque un train fantôme, que Rocrail arrête les trains et qu'il faut couper l'alimentation. L'unique session propre de 17 minutes fait figure d'exception, non de nouvelle norme.

Pourquoi l'ATtiny85 constitue une limite matérielle

Trois tentatives de correction du micrologiciel — condition de course, maintien en position, réglages de temporisation — n'ont pas résolu le problème. La raison semble tenir à l'architecture du composant plutôt qu'à un défaut du code. Le décodeur doit assurer simultanément deux tâches critiques à la microseconde, et le circuit n'offre aucun moyen de les séparer :

- **Un seul cœur à 8 MHz.** La génération des impulsions servo et la mesure de la durée des bits DCC exigent toutes deux une temporisation précise, et un seul processeur doit s'en charger. Rien ne s'exécute en parallèle.
- **Un compteur unique et partagé.** La bibliothèque servo produit ses impulsions à partir du Timer1, qui est aussi le seul compteur d'une résolution suffisante disponible pour le reste du programme. Les deux tâches se disputent la même ressource.
- **Les interruptions se bloquent mutuellement.** Sur AVR, tant qu'une routine d'interruption s'exécute, les autres sont mises en attente. Le DCC distingue un bit « 1 » d'un bit « 0 » par une demi-impulsion d'environ 58 μ s contre 100 μ s ou plus ; si une interruption du compteur servo retarde la mesure d'un front au-delà de sa fenêtre valide, le bit est mal interprété et la trame entière est rejetée. Un servo attaché, qui impulse toutes les 20 ms, garantit des occasions récurrentes de collision de ce type.
- **Aucun moyen matériel de les découpler.** C'est la différence décisive avec le Raspberry Pico. Le PIO du RP2040 est une petite machine à états programmable qui mesure les fronts du signal dans un périphérique dédié, indépendamment de ce que fait le processeur : une trame ne peut pas être manquée parce que le CPU était occupé ailleurs. L'ATtiny85 ne possède pas d'équivalent, si bien que cette concurrence ne peut pas être éliminée par le logiciel.

C'est la même classe de problème que le PIO a résolu définitivement sur les décodeurs de solénoïdes. Cela reste toutefois l'explication la plus probable et non une certitude établie : c'est précisément pourquoi le plan de mesure de la section 7 vise à la distinguer d'un défaut matériel ou de câblage propre à certaines cartes avant toute décision de remplacement.

7. Prochaines étapes

Séquence retenue, une variable à la fois :

1. Flasher le micrologiciel de servo v0.9 sur les huit décodeurs ATtiny85. Seuls deux sont actuellement à jour, si bien qu'une défaillance sur les six autres n'apprend rien — elle pourrait simplement relèver du décalage de position que la v0.9 corrige déjà. DCC_ADDRESS doit être réglé carte par carte (167–174).
2. Enchaîner plusieurs sessions dans les conditions actuelles (trois trains, sens unique) en relevant deux chiffres par session : le nombre de minutes avant la première défaillance de servo, et l'adresse concernée.
 - Si les défaillances se concentrent sur certaines cartes, la cause est probablement matérielle ou liée au câblage de ces unités, donc réparable.
 - Si elles se répartissent uniformément sur les huit, c'est la plateforme ATtiny85 qui est le facteur limitant.
 - Le délai avant première défaillance détermine si l'exploitation automatique est envisageable dans la configuration actuelle.
3. Seulement ensuite, ajouter un quatrième train et autoriser la circulation bidirectionnelle. Cela supprime l'effet de masque qui dissimule aujourd'hui les défaillances sur aiguilles prises en talon et révélera la véritable distribution des défaillances — mais ce n'est un essai pertinent que sur un système censé y survivre.

7.1 Solution de repli

Si les décodeurs ATtiny85 ne peuvent être fiabilisés, l'alternative éprouvée consiste en un décodeur de servo sur Raspberry Pico, combinant la réception PIO déjà validée sur ce réseau et la commande de servos.

La contrainte est physique et non technique : les fils de commande des servos sont sensibles aux perturbations sur de longues distances, et l'avantage actuel des cartes ATtiny85 tient précisément à ce que chacune se trouve immédiatement à côté de son servo. Une carte Pico multi-voies allongerait nécessairement certains câblages. L'emplacement des cartes, le nombre de voies par carte et, éventuellement, des variantes RP2040 plus compactes conservant le PIO devront être mis en balance avec cet inconvénient.

8. Livrables produits

Livable	Objet
Analyseur DCC (Pico + 6N137, réception PIO)	Mesure de référence sur le bus DCC
dcc_pulse.pio	Programme PIO de mesure d'impulsions, commun à l'analyseur et au décodeur
dcc_logger.py	Capture série horodatée, une instance par appareil
dcc_reconcile.py	Comparaison automatisée des ordres Rocrail et des trames sur le bus
rocrail_CAN_TCP_GW_v2.1.0.ino	Passerelle avec compteurs de pertes, journalisation des acquittements et reprise en boucle fermée
pico_dcc_decoder_v0.7.ino	Décodeur solénoïde avec réception PIO et file de commandes
ATtiny-DCC-Accessory-Decoder-v0.9.ino	Décodeur de servo avec correction de la course et maintien en position

9. Conclusions

Le défaut initial n'était pas une anomalie unique mais **six défauts indépendants répartis sur quatre étages**, dont plusieurs produisaient des symptômes identiques. C'est pourquoi le problème a si longtemps résisté au diagnostic : corriger l'un d'eux laissait le symptôme apparemment inchangé, et une session d'essai isolée ne permettait pas de les distinguer.

L'étape méthodologique décisive a été la construction d'un instrument de mesure indépendant — l'analyseur — et de l'outillage de réconciliation associé. Dès lors que chaque maillon pouvait être vérifié séparément, chaque étage a pu être mis hors de cause ou incriminé sur pièces plutôt que par déduction, et les deux modes de défaillance restants ont pu être distingués par un marqueur unique et fiable : la présence ou l'absence de l'acquiescement du Gleisbox.

Les pertes d'ordres en amont des décodeurs ont été éliminées. Subsiste un problème unique et bien caractérisé, circonscrit à huit décodeurs de servos, assorti d'un plan de mesure clair pour déterminer s'il est corrigible par logiciel, réparable matériellement sur certaines unités, ou s'il impose un changement de plateforme.

BROUILLON