



Août 2026

Martín LUZARDO

Tuteur académique : Amer BAGHDADI

Tuteur entreprise : Sébastien BASSET

Prototypage et intégration sur des systèmes embarqués basse consommation pour la transmission vidéo en direct

Rapport de Stage de fin d'études

Systèmes Embarqués

Avril 2026 – Septembre 2026



IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

HAIVISION

© 2026 - *MARTÍN LUZARDO*
ALL RIGHTS RESERVED.

INSTITUT MINES TÉLÉCOM ATLANTIQUE
BREST, BRETAGNE, FRANCE
HAIVISION
RENNES, BRETAGNE, FRANCE

Prototypage et intégration sur des systèmes embarqués basse consommation pour la transmission vidéo en direct

ABSTRACT

Haivision, acteur reconnu de la diffusion vidéo en direct, innove pour répondre aux exigences des réseaux 5G. Dans ce contexte, l'objectif est de développer des plateformes électroniques basse consommation. Intégrée à l'équipe Embedded Systems, la mission s'inscrit principalement dans le développement logiciel embarqué.

Le projet principal porte sur la configuration et l'optimisation de systèmes Linux via Yocto et Buildroot, avec l'intégration de mécanismes critiques tels que le démarrage sécurisé (Secure Boot) et les mises à jour logicielles à distance (OTA/RAUC). En parallèle, la mission s'élargit pour apporter un support transversal sur d'autres projets de l'entreprise, en y déployant ces mêmes implémentations logicielles.

Les activités techniques ont inclus le développement en C pour l'encodage vidéo, la gestion de pipelines (GStreamer, FFmpeg), et l'intégration des protocoles de transport (SRT, SST). Les travaux réalisés ont permis de valider une architecture logicielle robuste et redondante (A/B + Rescue), de sécuriser la chaîne de confiance matérielle via des images FIT signées RSA sans dégradation du temps d'amorçage, et d'automatiser entièrement la génération des firmwares via des pipelines CI/CD sous Azure DevOps et TeamCity. Ces livrables garantissent la fiabilité, la cyberrésilience (conformité anticipée CRA) et les performances opérationnelles des futurs produits stratégiques d'Haivision.

Mots-clés : Linux Embarqué, Yocto, Buildroot, Secure Boot, RAUC, 5G, GStreamer, SRT, SST, Haivision.

Table des matières

1	INTRODUCTION	1
1.1	Contexte global	1
1.2	Problématique d'ingénierie et cadre de la mission	2
1.3	Enjeux pour l'entreprise	3
1.4	Projet professionnel	3
2	PRÉSENTATION DE LA STRUCTURE D'ACCUEIL ET DES ENJEUX	5
2.1	Haivision et son positionnement sur le marché	5
2.2	Les enjeux sociétaux et environnementaux de la vidéo embarquée	7
2.3	Organisation interne et environnement de travail	8
2.4	Présentation matérielle du projet	9
3	CONCEPTION ET RÉALISATION DE LA MISSION	11
3.1	Méthodologie et gestion de projet	11
3.2	Architecture et personnalisation du système d'exploitation bas niveau	12
3.3	Implémentation du système de mise à jour à distance robuste (RAUC)	16
3.4	Sécurisation du système : Implémentation du Secure Boot	18
3.5	Automatisation et chaîne d'intégration continue (CI/CD)	21
4	ÉVALUATION, PRISE DE RECUL ET IMPACT DE LA MISSION	24
4.1	Validation technique et performances de la plateforme	24
4.2	Impact sur l'optimisation énergétique	26
4.3	Optimisation des flux de travail et agilité de l'équipe	26
4.4	De la R&D à l'industrialisation : ROI et avantage concurrentiel	27

5	CONCLUSION ET PERSPECTIVES	29
5.1	Bilan technique et scientifique	29
5.2	Perspectives industrielles et techniques	30
5.3	Bilan personnel et montée en compétences	30
	REFERENCES	32
	ANNEXE — UTILISATION DE L'IA	33

Table des figures

2.4.1 Plateforme Makito ONE (Architecture x86 / Buildroot)	10
3.4.1 Schéma de la Chaîne de confiance	20
3.5.1 Architecture des pipelines d'intégration continue (CI/CD)	22
3.5.2 Flux de déploiement et de mise à jour du firmware	23

Remerciements

Je souhaite tout d'abord exprimer ma profonde gratitude envers la Faculté d'Ingénierie de l'UDELAR et l'ensemble de ses enseignants, ainsi qu'envers l'ANII pour son précieux soutien financier et institutionnel tout au long de ces deux dernières années.

J'adresse également mes sincères remerciements à IMT Atlantique, à sa direction et à son corps professoral pour la qualité de l'enseignement reçu.

J'adresse des remerciements tout particuliers à mon tuteur d'entreprise, Sébastien BASSET, pour sa confiance et son encadrement tout au long de cette phase d'industrialisation. Je remercie également mon précédent tuteur, Nicolas DHOLLANDE, pour son accompagnement et la transmission de ses connaissances lors de mes premiers travaux au sein de l'équipe Advanced R&D.

Je remercie sincèrement mon tuteur académique d'IMT Atlantique, Amer BAGHDADI, pour son suivi régulier, ses remarques constructives et ses conseils lors de l'élaboration de ce travail de fin d'études.

Je souhaite plus largement remercier l'ensemble des membres des équipes d'Haivision pour leur accueil, leur bienveillance et la richesse des échanges techniques partagés au quotidien.

Enfin, j'adresse une pensée toute particulière à ma famille, mes proches et mes amis, en Uruguay comme en France, pour leur soutien inconditionnel, leurs encouragements et leur présence constante tout au long de mon parcours.

1

Introduction

1.1 CONTEXTE GLOBAL

AU COURS DE LA DERNIÈRE DÉCENNIE, la transmission vidéo en direct haute définition à très faible latence est devenue un composant critique de nombreuses infrastructures stratégiques. Des réseaux de contribution audiovisuelle pour les grands événements sportifs et journalistiques jusqu'aux applications de défense, de sécurité publique et de télémédecine, la demande pour des flux vidéo ininterrompus, fiables et haute fidélité n'a cessé d'augmenter.

L'émergence des réseaux cellulaires de cinquième génération (5G) et la modernisation des réseaux IP ont profondément transformé les architectures de diffusion. Elles permettent désormais le déploiement d'équipements vidéo nomades et embarqués directement sur le terrain, s'affranchissant des lourdes infrastructures satellitaires traditionnelles. Dans ce

paysage technologique en mutation rapide, la qualité d'expérience (QoE) repose non seulement sur l'efficacité des algorithmes de compression et de transport vidéo (tels que le protocole SRT - *Secure Reliable Transport*), mais également sur la stabilité, la réactivité et l'efficacité énergétique des plateformes matérielles et matérielles-logicielles (*firmwares*) qui exécutent ces traitements en temps réel.

1.2 PROBLÉMATIQUE D'INGÉNIERIE ET CADRE DE LA MISSION

Ce stage de fin d'études s'inscrit dans la continuité directe de mon parcours au sein d'Haivision, faisant suite à mon stage de l'année précédente et à ma récente année en contrat de professionnalisation. Cette ultime phase marque une transition stratégique dans mon évolution au sein de l'entreprise : initialement rattaché à l'équipe *Advanced R&D* pour des travaux exploratoires, j'ai désormais rejoint l'équipe *Embedded Systems*. Ce changement d'équipe s'accompagne d'un recentrage technique vers l'industrialisation, avec un fort accent mis sur le développement Linux embarqué, la configuration de drivers et la mise en place de chaînes d'intégration continue (CI/CD) transverses pour les différents projets de l'entreprise.

Dans le cadre du développement de ces produits de nouvelle génération, l'un des défis majeurs consiste donc à assurer la transition entre des prototypes logiciels et des architectures matérielles cibles prêtes pour un déploiement à grande échelle. La problématique centrale de ce travail d'ingénierie s'articule ainsi autour de la question suivante :

Comment concevoir et déployer une architecture logicielle Linux embarquée robuste, sécurisée et automatisée, capable de garantir l'intégrité du système (Secure Boot), d'assurer des mises à jour à distance fiables (RAUC) et d'optimiser l'efficacité opérationnelle des plateformes matérielles cibles ?

Répondre à cette problématique nécessite d'intervenir à la frontière entre le matériel et le logiciel bas niveau : personnalisation de la distribution Linux embarquée via des outils de génération de système (*Yocto / Buildroot*), maîtrise de la chaîne d'amorçage (*U-Boot*), développement et intégration de drivers, et implémentation de mécanismes cryptographiques de confiance.

1.3 ENJEUX POUR L'ENTREPRISE

Pour Haivision, leader mondial des solutions de contribution et de distribution vidéo à faible latence, les enjeux associés à cette mission sont à la fois techniques, opérationnels, stratégiques et réglementaires :

- **Garantie de l'intégrité et de la sécurité logicielle** : Les équipements embarqués étant déployés dans des environnements parfois critiques ou exposés sur des réseaux publics, la protection contre la corruption de firmware et les injections de code malveillant (*cyberattaques*) est primordiale.
- **Conformité réglementaire européenne (Cyber Resilience Act - CRA) [4]** : L'implémentation du démarrage sécurisé (*Secure Boot*) et des mécanismes d'authentification cryptographique s'inscrit directement dans la préparation à la nouvelle réglementation européenne sur la cyberrésilience (CRA). Ce cadre législatif imposera très prochainement des normes strictes de sécurité dès la conception et de traçabilité logicielle pour tous les produits connectés commercialisés dans l'Union Européenne.
- **Maintien en condition opérationnelle (MCO) et réduction du SAV** : La mise en place d'un système de mise à jour à distance fiable avec mécanisme de repli automatique permet d'assurer le déploiement rapide des correctifs de sécurité exigés par le CRA, d'éviter l'immobilisation des boîtiers sur le terrain et de réduire drastiquement les coûts d'intervention de maintenance.
- **Agilité du cycle de développement** : L'automatisation de la génération et de la signature des images firmware via une chaîne d'intégration continue (CI/CD) accélère le déploiement des fonctionnalités tout en sécurisant la qualité logicielle livrée aux équipes de validation.

1.4 PROJET PROFESSIONNEL

Cette mission de fin d'études consolide mon projet professionnel et s'inscrit dans le prolongement de mon double cursus académique. Bien que les missions qui m'ont été confiées soient principalement orientées vers l'ingénierie des systèmes embarqués (en lien direct avec la TAF SEH – *Systèmes Embarqués et Hétérogènes* d'IMT Atlantique), les

solides bases en télécommunications acquises lors de ma formation en Ingénierie des Systèmes de Communication à l'Universidad de la República (UDELAR) et de la TAF STAR (*Systèmes de Transmission, Architectures et Réseaux*) restent étroitement liées aux produits développés chez Haivision. Elles constituent un avantage essentiel pour appréhender les problématiques de transport vidéo, facilitant ainsi une compréhension globale des équipements et de leurs contraintes réseau.

Passionné par l'ingénierie des systèmes embarqués, la transition de la R&D vers l'équipe *Embedded Systems* me permet de confronter directement ces compétences scientifiques croisées aux exigences strictes de l'industrialisation. En passant de l'exploration technologique au développement de solutions de production, cette expérience complète ma formation et prépare directement mon insertion sur le marché de l'emploi en tant qu'ingénieur en systèmes embarqués.

2

Présentation de la structure d'accueil et des Enjeux

2.1 HAIVISION ET SON POSITIONNEMENT SUR LE MARCHÉ

2.1.1 PRÉSENTATION DE L'ENTREPRISE ET LEADERSHIP TECHNOLOGIQUE

Fondée en 2004 et basée conjointement à Montréal et Chicago, Haivision est l'un des principaux fournisseurs mondiaux de solutions de vidéo en temps réel et de collaboration visuelle. L'entreprise s'est imposée comme un acteur incontournable sur le marché de la contribution et de la distribution vidéo sur IP, avec une présence internationale marquée par des centres de R&D, des bureaux commerciaux et des services de support déployés en Amérique, en Europe et en Asie.

L'une des forces majeures d'Haivision réside dans son modèle de développement

totallement intégré : l'ensemble de la chaîne de valeur, depuis la conception initiale des cartes électroniques et du matériel jusqu'à la production finale, en passant par le développement du logiciel bas niveau et des couches applicatives, est géré en interne. Cette maîtrise de bout en bout favorise une synergie étroite entre les équipes de R&D et offre un accès direct et privilégié aux compétences des différents départements (matériel, FPGA, logiciel embarqué, validation).

L'excellence technologique d'Haivision a été consacrée par plusieurs récompenses prestigieuses, notamment de multiples *Emmy Awards* décernés par la *National Academy of Television Arts and Sciences* pour ses innovations pionnières dans le domaine du transport vidéo à faible latence. Haivision se distingue notamment par la maîtrise de deux technologies de transport majeures :

- **Le protocole SRT [9] (*Secure Reliable Transport*)** : Protocole open-source pionnier développé à l'origine par Haivision, le SRT optimise la diffusion vidéo en direct sur des réseaux imprévisibles (comme Internet public). Il combine le chiffrement AES, la récupération dynamique de perte de paquets et la prévention de la gigue pour garantir la qualité du flux tout en simplifiant le franchissement des pare-feux.
- **Le protocole propriétaire SST [5] (*Safe Stream Transport*)** : Récompensé par deux *Emmy Awards*, le SST est la technologie propriétaire d'Haivision conçue pour dépasser les limites des réseaux cellulaires inégaux. Il permet l'agrégation de liens (*network bonding*) et la transmission vidéo de qualité télévision (*broadcast-grade*) à travers des réseaux hétérogènes et fortement instables (3G/4G/5G, Wi-Fi, satellite, LAN et Internet public).

2.1.2 SECTEURS D'ACTIVITÉ ET ENVIRONNEMENT CONCURRENTIEL

Les solutions développées par Haivision s'articulent autour de quatre piliers stratégiques majeurs :

- **Contribution vidéo en direct** : Solutions d'encodage et de décodage matériel à ultra-faible latence adaptées aux architectures de production à distance (*REMI*) et multi-caméras sur réseaux cellulaires et IP.
- **Défense, sécurité publique et ISR (*Intelligence, Surveillance, and Reconnaissance*)** : Équipements durcis et optimisés pour diffuser de la vidéo

critique à faible débit avec des exigences de sécurité de niveau militaire.

- **Salles de commandement et murs vidéo (*Command 360*)** : Plateformes logicielles et matérielles sécurisées permettant la prise de décision tactique en temps réel au sein des centres d'opérations stratégiques.
- **Distribution IPTV et entreprise** : Architectures de diffusion vidéo sécurisées sur réseaux d'entreprise sans saturation de la bande passante.

Dans cet écosystème, Haivision évolue face à une concurrence spécialisée. L'avantage concurrentiel d'Haivision repose sur la performance de ses moteurs d'encodage matériel, l'efficacité de ses protocoles propriétaires et open-source (SRT), ainsi que sur une maîtrise complète de la chaîne logicielle bas niveau garantissant une fiabilité éprouvée sur le terrain.

2.2 LES ENJEUX SOCIÉTAUX ET ENVIRONNEMENTAUX DE LA VIDÉO EMBARQUÉE

2.2.1 ENJEU SOCIÉTAL ET SÉCURITÉ DES DONNÉES

Dans une société contemporaine hyperconnectée, les flux vidéo véhiculent souvent des informations d'une extrême sensibilité : retransmissions d'opérations de maintien de l'ordre, données de reconnaissance militaire ISR, flux médicaux de télémedecine ou visio-conférences gouvernementales. La compromission d'un équipement vidéo embarqué exposé sur un réseau public ou cellulaire représente un risque cyber majeur (usurpation d'identité, interception de flux, injection de code malveillant).

À ce titre, la mise en place d'une chaîne de confiance matérielle et logicielle (*Secure Boot*) n'est pas uniquement une contrainte technique, mais une responsabilité sociétale. En garantissant l'intégrité absolue du firmware et en empêchant l'exécution de logiciels non autorisés, mon travail participe directement à la protection des données sensibles et à la résilience des infrastructures critiques contre les cyberattaques, s'inscrivant pleinement dans l'esprit du nouveau cadre réglementaire européen (*Cyber Resilience Act*).

2.2.2 ENJEU ENVIRONNEMENTAL ET SOBRIÉTÉ ÉNERGÉTIQUE

Les équipements vidéo embarqués sont par nature soumis à des contraintes thermiques et énergétiques strictes (fonctionnement sur batterie, refroidissement passif, déploiement

nomade). L'empreinte carbone et l'efficacité énergétique des infrastructures télécoms/5G constituent désormais un enjeu environnemental prioritaire.

Mon intervention au niveau du système d'exploitation Linux bas niveau (via Yocto et Buildroot) permet d'alléger l'OS en supprimant les services superflus et d'optimiser l'interaction avec le matériel cibles. En réduisant l'empreinte mémoire, en minimisant les cycles CPU inutilement consommés et en affinant la gestion des drivers matériels, cette démarche d'optimisation logicielle contribue directement à limiter la consommation électrique globale des boîtiers embarqués et à prolonger leur durée de vie opérationnelle sur le terrain.

2.3 ORGANISATION INTERNE ET ENVIRONNEMENT DE TRAVAIL

2.3.1 L'ÉQUIPE EMBEDDED SYSTEMS

Initialement intégré à l'équipe *Advanced R&D* pour des travaux d'exploration et de prototypage, j'ai rejoint durant cette mission la nouvelle équipe *Embedded Systems*. Cette structure dédiée a été créée pour porter l'industrialisation logicielle et assurer une passerelle opérationnelle entre la R&D, les équipes matérielles (électronique, FPGA, bancs de test) et les développeurs d'applications vidéo de haut niveau (protocoles de streaming, interfaces utilisateur, traitement de flux).

Mon rôle au sein de l'équipe *Embedded Systems* consiste à fournir des plateformes logicielles bas niveau (*Board Support Packages* - BSP, noyaux Linux personnalisés, drivers et chaînes d'amorçage sécurisées) robustes, standardisées et directement prêtes pour la production industrielle.

2.3.2 COLLABORATIONS PLURIDISCIPLINAIRES ET MODES DE TRAVAIL

Le développement de produits embarqués exige une synergie constante entre différents corps de métier :

- **Interface Électronique & FPGA** : Validation des schémas électroniques, interfaçage des IP-cores et développement des drivers Kernel spécifiques.
- **Interface Logiciel Applicatif** : Mise à disposition d'APIs et de systèmes d'exploitation stables pour les équipes de traitement vidéo applicatif.

- **Interface Assurance Qualité (QA) & Produit** : Automatisation de la génération des images via nos pipelines CI/CD pour faciliter les cycles de tests continus et garantir la qualité des livrables.

L'organisation du travail s'appuie sur une gestion de flux continu sous forme de tableau Kanban, particulièrement adaptée aux contraintes de la R&D et de l'ingénierie bas niveau. Le suivi et la documentation des tâches sont centralisés sur la suite Atlassian (Jira, Confluence), tandis que la gestion des versions de code repose sur Git/Bitbucket. L'automatisation des compilations et de l'intégration continue (CI/CD) est quant à elle orchestrée via les plateformes Azure DevOps (ADO) et TeamCity.

2.4 PRÉSENTATION MATÉRIELLE DU PROJET

Les travaux menés au cours de ce stage s'articulent autour de deux plateformes matérielles clés de nouvelle génération chez Haivision, associées à deux outils de génération de système d'exploitation distincts (*Yocto* pour les architectures ARM, *Buildroot* pour les architectures x86) :

- **Écosystème Falkon FX & CellLink5G (Architecture ARM)** :
 - **Falkon FX** : Équipement compact, robuste et sans ventilateur, conçu pour l'encodage vidéo temps réel en environnement critique sur réseaux IP et cellulaires. Suite aux évaluations comparatives de performances d'encodage vidéo menées lors de mon stage de l'année précédente, la plateforme s'appuie sur le SoC NXP i.MX95 [7]. Ce processeur garantit un encodage H.264/HEVC jusqu'en 1080p à 60 i/s avec une consommation électrique minimale.
 - **CellLink5G** : Module d'antenne déportée et de transmission RF réseau conçu pour fonctionner en synergie avec le Falkon FX. Contrairement au Falkon FX, le CellLink5G ne réalise pas de traitement ni d'encodage vidéo local ; il agit exclusivement comme un nœud de transmission et d'antenne intelligente. Par souci d'homogénéité matérielle et logicielle, il partage la même architecture. L'environnement logiciel bas niveau de cet écosystème ARM repose sur une distribution Linux épurée et personnalisée via **Yocto**.
- **Makito ONE [6] (Architecture x86 & ARM)** : Plateforme de transport vidéo de nouvelle génération conçue pour succéder aux encodeurs/décodeurs phares de la



Figure 2.4.1 – Plateforme Makito ONE (Architecture x86 / Buildroot)

gamme Makito. Architecture ultra-flexible et haute densité, le Makito ONE s'appuie sur une carte à processeur Intel x86 exécutant un OS Linux généré sous Buildroot :

- *Capacités vidéo* : Support multi-codec (H.264, HEVC et JPEG XS à très faible latence), traitement 4K UHD 4 : 2 : 2 10-bit et interfaces 12G-SDI ou IP SMPTE ST 2110 (jusqu'à 25 Gbps).
- *Gestion de châssis* : En parallèle de la carte x86 principale, je participe également au développement et à l'interfaçage logiciel du système de gestion de châssis articulé autour d'un microcontrôleur STM32MP157. Ce dernier assure le contrôle matériel bas niveau, la gestion de l'alimentation redondante et la supervision thermique.

Ces plateformes matérielles hétérogènes (x86 et ARM) constituent le socle d'application sur lequel l'ensemble des briques logicielles bas niveau développées pendant ce stage — démarrage sécurisé (*Secure Boot*), drivers matériels, mises à jour redondantes (*RAUC*) et pipelines d'intégration continue (*CI/CD*) — ont été implémentées et validées.

3

Conception et réalisation de la mission

Ce chapitre présente le cœur technique et scientifique de mes travaux d'ingénierie. Il détaille la méthodologie de développement, la conception des systèmes d'exploitation embarqués bas niveau sur architectures x86 et ARM, la mise en place d'une chaîne de confiance cryptographique complète (*Secure Boot*), la déploiement d'un mécanisme de mise à jour robuste et redondant (*RAUC*), ainsi que l'automatisation globale du cycle d'intégration continue.

3.1 MÉTHODOLOGIE ET GESTION DE PROJET

3.1.1 ORGANISATION DU TRAVAIL ET SUIVI EN FLUX CONTINU (KANBAN)

Afin de répondre aux contraintes du développement bas niveau embarqué, le suivi du projet s'appuie sur une démarche Kanban d'amélioration continue du flux de travail, intégrée au sein de la suite Atlassian (Jira et Confluence). Les tâches logicielles et

d'intégration sont suivies de manière transparente entre les équipes matérielles (FPGA, électronique) et logicielles.

Pour concilier l'agilité nécessaire au développement quotidien et la rigueur requise par la production industrielle, une distinction stricte a été établie au niveau du cycle de build :

- **Environnement de développement** : Utilisation d'un SDK externe partagé permettant aux développeurs de compiler rapidement leurs applications et drivers sans régénérer l'ensemble de l'image Linux.
- **Environnement de production** : Intégration de recettes Yocto/Buildroot dédiées basées sur des *tags* Git précis, garantissant une reproductibilité stricte des firmwares signés livrés en validation.

3.1.2 GESTION DES VERSIONS ET AUTOMATISATION (CI/CD)

Le code source des différentes briques (BSP, bootloaders, noyaux Linux, drivers) est versionné sous Git et hébergé sur Bitbucket. Les compilations croisées, les vérifications d'intégrité et la génération automatisée des bundles de mise à jour sont orchestrées de manière complémentaire via Azure DevOps et TeamCity, alimentant directement les dépôts de livrables internes.

3.2 ARCHITECTURE ET PERSONNALISATION DU SYSTÈME D'EXPLOITATION BAS NIVEAU

3.2.1 DÉCOHABITATION DES OUTILS DE BUILD : YOCTO ET BUILDROOT

Historiquement, l'environnement logiciel embarqué des équipements Haivision reposait principalement sur Buildroot, apprécié pour sa simplicité, son approche minimaliste et ses temps de génération linéaires. Toutefois, l'intégration des nouvelles architectures processeurs hétérogènes (ARM / microcontrôleurs) a nécessité une évolution stratégique vers le projet Yocto :

- **Transition vers Yocto [10] pour l'écosystème ARM (Falkon FX / CellLink5G – SoC NXP i.MX95)** : Bien que Buildroot convienne aux architectures simples, le passage à Yocto a été motivé par la nécessité de bénéficier du support officiel des fondeurs de semi-conducteurs (*silicon vendors* tels que NXP et STMicroelectronics).

Ces fabricants maintiennent et publient leurs couches BSP de référence (*Board Support Packages*, telles que `meta-freescale` ou `meta-arm`) directement sous forme de *layers* Yocto. L'adoption de Yocto permet ainsi de récupérer nativement les correctifs matériels, les accélérations graphiques/VPU, les modules du noyau spécifiques au traitement vidéo et l'intégration out-of-the-box des briques d'amorçage sécurisé (TF-A, U-Boot).

- **Maintien de Buildroot [2] pour l'architecture x86 (Makito ONE)** : Pour la carte processeur Intel x86 du Makito ONE, l'utilisation de Buildroot a été conservée. Elle permet de générer un système de fichiers racine (*RootFS*) épuré, déterministe et ultra-rapide au démarrage, parfaitement calibré pour exécuter les binaires applicatifs sur une cible PC embarquée standard sans la complexité de maintenance d'une distribution complète.

Le Tableau 3.2.1 synthétise l'analyse comparative et les justifications techniques ayant guidé cette répartition d'outils au sein des plateformes de l'entreprise.

3.2.2 STRATÉGIE DE PARTITIONNEMENT DU STOCKAGE

Afin de concilier la protection du système contre la corruption et la conservation des données d'exécution, la structure du stockage flash/eMMC a été normalisée. La répartition des partitions a été définie comme l'indique le Tableau 3.2.2.

3.2.3 CHOIX DE L'ALGORITHME DE SIGNATURE

Lors des phases initiales d'implémentation du *Secure Boot* sur la plateforme Makito ONE, l'utilisation de signatures ECDSA (*Elliptic Curve Digital Signature Algorithm*) avait été privilégiée afin d'optimiser la taille des clés et la rapidité d'exécution. Cependant, des échecs systématiques d'authentification lors du traitement par U-Boot ont été constatés, liés à des incompatibilités au niveau des pilotes d'accélération matérielle cryptographique.

Afin de sécuriser le calendrier d'industrialisation et de garantir la stabilité du système, le choix s'est porté sur un schéma de signature asymétrique RSA-2048/4096. J'ai ainsi mis en place une chaîne complète et fonctionnelle de signature et de vérification des images FIT (*fitImage*) sous U-Boot. Cette configuration répond pleinement aux exigences de

Table 3.2.1 – Analyse comparative entre Yocto Project et Buildroot chez Haivision

Critère d'évaluation	Yocto Project (Poky / BitBake)	Buildroot
Architecture cible	ARM (SoC NXP i.MX95 / STM32)	Intel x86
Plateformes Haivision	Falkon FX / CeilLink5G	Makito ONE
Support fondeur (BSP)	Natif et officiel (maintien direct par NXP, STMicroelectronics, ARM)	Manuel (nécessite l'intégration et l'adaptation manuelle des patches constructeur)
Philosophie	Générateur de distributions complètes (<i>Layer-based</i>)	Générateur de <i>RootFS</i> et <i>toolchains</i> minimalistes
Gestion des paquets	Paquets individuels (.deb, .rpm, .ipk) avec gestion fine des dépendances	Compilation globale monolithique, pas de gestionnaire de paquets dynamique
Temps de compilation	Élevé lors du build initial, optimisé ensuite via le cache partagé (<i>sstate-cache</i>)	Rapide et linéaire, mais nécessite souvent un <i>re-build</i> complet lors des modifications
Génération de SDK	Extrêmement modulaire (<i>extensible SDK</i> - eSDK) pour le développement applicatif	SDK monolithique basé sur la <i>toolchain</i> croisée
Flexibilité & Maintenance	Très forte via les <i>layers</i> (<i>meta-haivision</i> , <i>meta-nxp</i>) et <i>bbappends</i>	Configuration simple et centralisée via Kconfig (<i>menuconfig</i>)
Cas d'usage retenu	Nouvelles plateformes ARM complexes, briques sécurisées (TF-A/OP-TEE) et VPU	Systèmes embarqués dédiés x86 exigeant un démarrage rapide et déterministe

Table 3.2.2 – Schéma de partitionnement standardisé pour les plateformes cibles

Partition	Taille	Système de fichiers	Description / Usage
boot	64 Mo	FAT32	Scripts d'amorçage
system_a	4 Go	EXT4	Système d'exploitation principal (Slot A)
system_b	4 Go	EXT4	Système d'exploitation principal (Slot B)
rescue	128 Mo	EXT4	OS minimal de secours
log	1 Go	EXT4	Journaux système et traces d'audit
config	512 Mo	EXT4	Paramètres réseau et configuration produit
license	100 Mo	EXT4	Licences logicielles et identifiants uniques
data	2 Go	EXT4	Données applicatives utilisateur
extended	Reste	exFAT	Espace de stockage étendu / enregistrements

robustesse et de sécurité imposées pour la mise en production actuelle. L'intégration d'algorithmes plus légers ou avancés basés sur les courbes elliptiques (ECDSA) reste envisagée comme une perspective d'évolution à moyen terme, une fois le support matériel stabilisé par les constructeurs.

3.2.4 ARCHITECTURE DU ROOTFS ET CONTRAINTES DE MÉMOIRE RAM

Plusieurs architectures de système de fichiers ont été évaluées pour garantir l'intégrité du *RootFS* :

- **SquashFS + dm-verity / ext4 + dm-verity** : Offrait une vérification à la lecture (*read-on-demand*) bloc par bloc avec un niveau de sécurité optimal, tout en évitant l'empreinte en mémoire RAM.
- **RAMDisk (Initramfs) chargé en RAM** : Offrait l'avantage d'une cohérence d'architecture absolue avec les autres plateformes déjà déployées chez Haivision.

L'implémentation du RAMDisk a présenté un défi majeur sur les cartes à ressources contraintes (notamment le gestionnaire de châssis basé sur STM32/médiums embarqués limités à 512 Mo de RAM), où la taille de l'image risquait de saturer la mémoire vive. Pour surmonter cette contrainte tout en conservant la simplicité et la standardisation inter-plateformes, un travail d'optimisation et d'élagage de l'image Linux a permis de

réduire drastiquement l’empreinte du RootFS, rendant l’approche RAMDisk pleinement fonctionnelle.

Note sur les performances de chargement : Sur les cartes Intel (Makito ONE), la lenteur d’I/O sous U-Boot a nécessité de déléguer le chargement du RootFS à Slim Bootloader. Cette métrique a servi de point de comparaison pour les tests d’optimisation sous U-Boot sur le SoC i.MX95.

3.3 IMPLÉMENTATION DU SYSTÈME DE MISE À JOUR À DISTANCE ROBUSTE (RAUC)

3.3.1 ARCHITECTURE DE MISE À JOUR ET STRATÉGIE DE REPLI

Pour éviter tout risque de verrouillage définitif du système (*bricking*) lors d’une coupure d’alimentation ou d’un échec de mise à jour sur le terrain, l’agent RAUC [8] (*Robust Auto-Update Controller*) a été intégré au système Linux bas niveau.

La conception du schéma de partitionnement a suivi une démarche itérative guidée par les contraintes opérationnelles et l’expérience utilisateur :

1. **Approche initiale (Main + Rescue) et ses limites** : Dans un premier temps, une architecture asymétrique à deux partitions (`main` et `rescue`) a été évaluée. Ce modèle présentait l’avantage d’optimiser l’espace de stockage flash en conservant une partition de secours très compacte. Cependant, ce schéma imposait une contrainte majeure : pour appliquer une mise à jour, le système devait obligatoirement redémarrer sur l’environnement `rescue`, procéder à l’écriture de la nouvelle image sur la partition principale, puis redémarrer une seconde fois. Cette séquence interrompait prématurément les services et ne permettait pas de lancer la mise à jour directement depuis l’application métier d’Haivision, ni de suivre sa progression en direct (l’application n’étant exécutée que sur l’environnement principal).
2. **Transition vers une architecture symétrique A/B (*Seamless Update*)** : Afin de répondre aux exigences de disponibilité des équipements de diffusion vidéo, le choix s’est porté sur un partitionnement symétrique à double slot (**Slot A** et **Slot B**). Cette

structure permet d'effectuer les mises à jour en tâche de fond (*in-place update*) directement depuis le système actif et l'application métier, offrant un suivi précis du pourcentage d'avancement sans interrompre le fonctionnement opérationnel de la carte. Le basculement vers le nouveau firmware ne nécessite alors qu'un seul redémarrage déterministe.

3. **Conservation de la partition Rescue pour une résilience maximale (A/B + Rescue)** : Plutôt que de supprimer l'environnement de secours au profit du modèle A/B standard, la partition `rescue` a été conservée comme troisième niveau de sécurité. Si une défaillance critique, une corruption matérielle ou des erreurs logicielles successives rendent inopérants à la fois le Slot A et le Slot B, le bootloader U-Boot (pilote par le *Watchdog*) bascule automatiquement sur la partition `rescue`. Cet environnement minimaliste autonome permet de maintenir l'accès réseau ou console, d'exécuter un diagnostic et de réinstaller un firmware valide via le réseau ou une clé USB d'installation.

3.3.2 SÉCURISATION DES CERTIFICATS ET CLÉS D'UPDATE

Le démon RAUC s'appuie sur OpenSSL pour vérifier l'authenticité des bundles de mise à jour avant toute écriture sur la mémoire flash. Le certificat public de validation (`rauc.cert`) est stocké sur la partition `rootfs` en lecture seule. Cette partition est intégrée au sein de la `fitImage` et se trouve donc signée, garantissant ainsi l'intégrité et la sécurité du certificat. Les clés de mise à jour (*Update Keys*) sont signées et leur intégrité est directement vérifiée par U-Boot [3] lors du démarrage du noyau.

3.3.3 STRUCTURE ET PACKAGING DES BUNDLES DE FIRMWARE

Les bundles générés sous forme d'archives signées RAUC encapsulent l'ensemble des composants nécessaires au rafraîchissement complet d'une carte :

`rauc-bundle.raucb`

<code>main.fit</code>	(<code>fitImage</code> qui contient le <code>rootfs</code> , <code>kernel</code> et <code>dtb</code>)
<code>bootscript.fit</code>	(Script d'amorçage U-Boot)
<code>sbl.bin</code>	(Binaire Bootloader)
<code>appli.bin</code>	(Application entreprise)

3.3.4 GESTION DE L'ENVIRONNEMENT U-BOOT ET PARTITIONNEMENT BRUT

Initialement, le stockage des variables d'environnement U-Boot était prévu sur la mémoire SPI Flash. Cependant, des instabilités matérielles lors des accès en lecture/écriture sur la SPI ont conduit à abandonner cette option au profit d'une partition brute (*raw partition*) dédiée sur le stockage eMMC/Flash principal.

De plus, l'écriture directe dans l'environnement depuis U-Boot présentant des risques de corruption lors de coupures d'alimentation, la stratégie a été modifiée :

- Les écritures dans l'environnement U-Boot sont exclusivement réalisées depuis l'espace utilisateur (*User Space*) de Linux via les utilitaires `fw_printenv` / `fw_setenv`.
- Les compteurs d'essais intégrés à RAUC ont été désactivés au niveau du bootloader. En cas de défaillance matérielle ou de plantage du démarrage, un *Watchdog* matériel réinitialise la carte et bascule directement le démarrage sur la partition de secours (*Rescue Partition*).

3.3.5 STOCKAGE ET PROTECTION DU CERTIFICAT DE VÉRIFICATION RAUC

La sécurité du système de mise à jour RAUC repose sur la protection du certificat public (`rauc.cert`) utilisé pour vérifier la signature des bundles d'installation (`.raucb`). Un stockage du certificat sur une partition modifiable présentait un risque d'usurpation ou de corruption.

Pour résoudre cette problématique, le certificat `rauc.cert` a été directement intégré au sein du *RootFS*. Comme le *RootFS* est lui-même encapsulé et signé au sein de la *fitImage* (validée par la BootROM et U-Boot lors du démarrage), l'intégrité et l'authenticité du certificat RAUC sont implicitement garanties par la chaîne de confiance matérielle (*Root of Trust*).

3.4 SÉCURISATION DU SYSTÈME : IMPLÉMENTATION DU SECURE BOOT

3.4.1 CHAÎNE DE CONFIANCE (*CHAIN OF TRUST*) ET ANCRAGE MATÉRIEL

L'implémentation du démarrage sécurisé repose sur une chaîne de confiance ininterrompue depuis le composant matériel jusqu'à l'application métier, comme l'illustre la Figure 3.4.1 :

1. **Racine de confiance matérielle (*Root of Trust*)** : Les clés publiques de hachage sont brûlées de manière irréversible dans les fusibles matériels (*eFuses*) du processeur.
2. **Étape d'amorçage primaire (*BootROM / TF-A [1]*)** :
 - Sur l'architecture ARM, la BootROM valide le binaire **TF-A (BL2)**, qui vérifie à son tour l'image **FIP** (*Firmware Image Package*) englobant U-Boot et l'environnement d'exécution sécurisé **OP-TEE**.
 - Sur l'architecture x86, la ROM du processeur valide le **Slimbootloader**, qui initialise la mémoire et passe la main à U-Boot.
3. **Amorçage du Noyau** : U-Boot valide l'intégrité cryptographique des images FIT avant leur exécution.

3.4.2 GÉNÉRATION, SIGNATURE ET VÉRIFICATION DES FIT IMAGES

Pour l'amorçage du noyau Linux et de son arbre de périphériques, la technologie des images **FIT** (*Flattened Image Tree*) a été retenue. Une FIT image regroupe dans un conteneur unique le noyau (`kernel.bin`), le Device Tree (`dtb`) et l'image du ramdisk (`rootfs`).

Chaque section du fichier FIT est signée lors du build à l'aide d'une clé privée RSA. Lors de la phase de boot, U-Boot utilise la clé publique RSA intégrée à sa propre structure de données (elle-même validée par le bootloader de niveau inférieur) pour vérifier la signature des hachages. Si la moindre modification non autorisée est détectée dans l'image FIT, l'amorçage est immédiatement interrompu.

3.4.3 PROCÉDURE DE PREMIÈRE INSTALLATION (*FIRST INSTALL*)

Pour le provisionnement initial des cartes vierges en usine ou en sortie de chaîne de fabrication, deux procédures de première installation sécurisées ont été développées :

- **Cartes x86 (Makito ONE)** : Flashage de base via un programmeur Dediprog (Slimbootloader + U-Boot), suivi d'une installation automatisée du système complet via une clé USB dédiée d'installation.
- **Cartes ARM i.MX95 (Falkon FX / CellLink5G)** : Déploiement via les outils DFU (*Device Firmware Upgrade*) de NXP pour installer U-Boot, suivi de l'utilisation de la

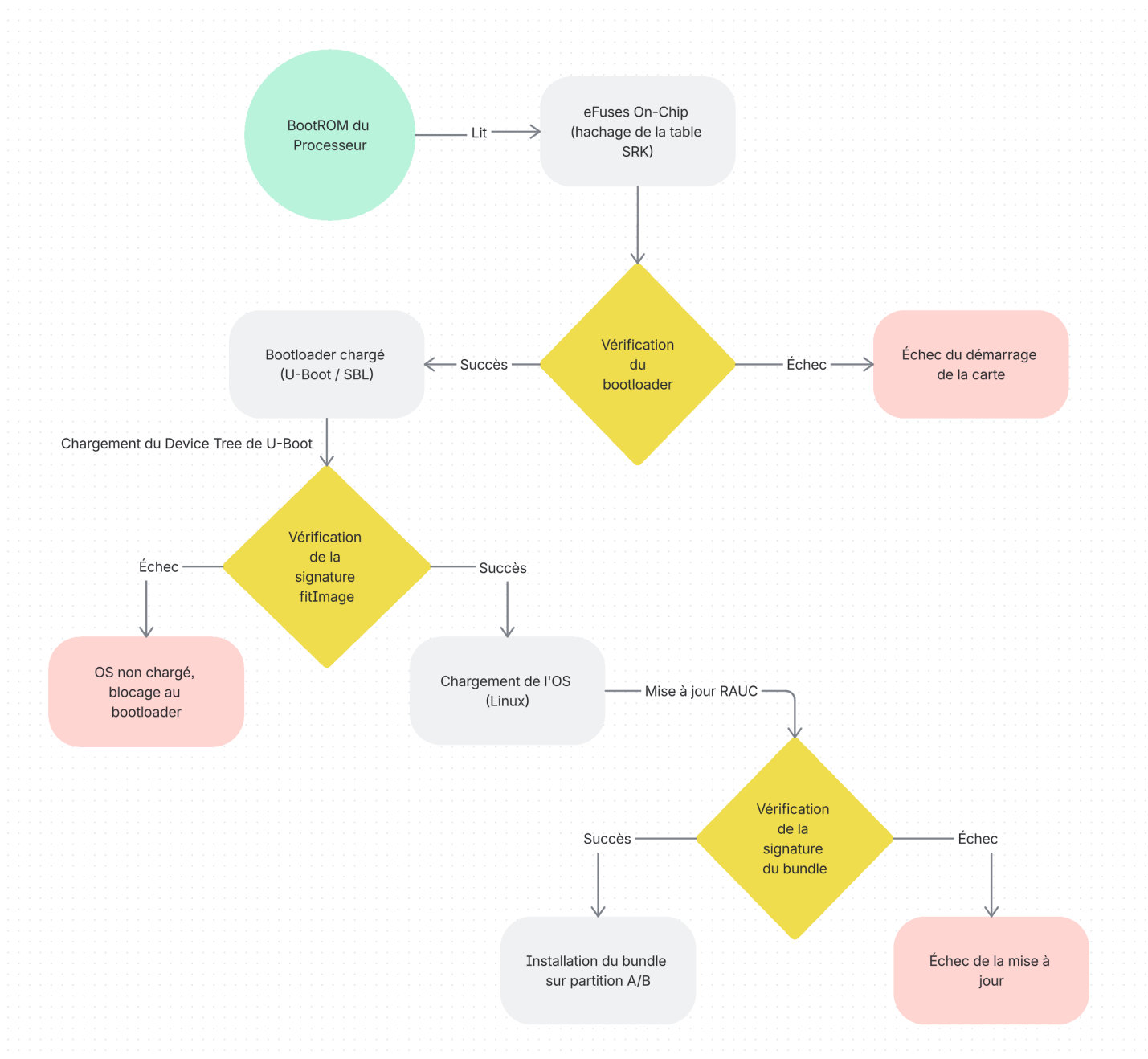


Figure 3.4.1 – Schéma de la Chaîne de confiance

clé USB d'installation pour déployer l'ensemble des partitions et le firmware cible.

3.5 AUTOMATISATION ET CHAÎNE D'INTÉGRATION CONTINUE (CI/CD)

3.5.1 PIPELINES DE GÉNÉRATION GÉNÉRIQUES (ADO / TEAMCITY)

Afin de garantir que chaque modification apportée sur Bitbucket produise une image valide et testable, des pipelines d'intégration continue ont été créés sous Azure DevOps et TeamCity.

Le cycle d'exécution d'un pipeline comprend les étapes automatisées suivantes (représentées sur la Figure 3.5.1) :

1. **Extraction et résolution des dépendances** : Récupération des recettes Yocto/Buildroot et déclenchement des requêtes d'extraction basées sur les *tags* de version de l'application métier et des bitstreams FPGA.
2. **Compilation croisée du système** : Génération isolée dans des conteneurs d'intégration pour garantir l'étanchéité de l'environnement de build.
3. **Signature cryptographique** : Récupération sécurisée des clés de signature via un coffre-fort de clés (*HSM*), signature des images FIT (Kernel/DeviceTree) et du binaire U-Boot.
4. **Génération des livrables** : Création simultanée de l'image d'installation initiale y compris l'image du bundle de mise à jour RAUC (*.raucb*).
5. **Publication et traçabilité** : Stockage des artefacts sur serveur de fichiers interne et journalisation des hachages pour contrôle d'audit.

Cette automatisation complète supprime les opérations de compilation manuelle, garantit la traçabilité intégrale des firmwares livrés et prépare l'industrialisation à grande échelle des nouvelles gammes d'équipements Haivision, tout en s'articulant avec le processus global de mise à jour logicielle présenté en Figure 3.5.2.

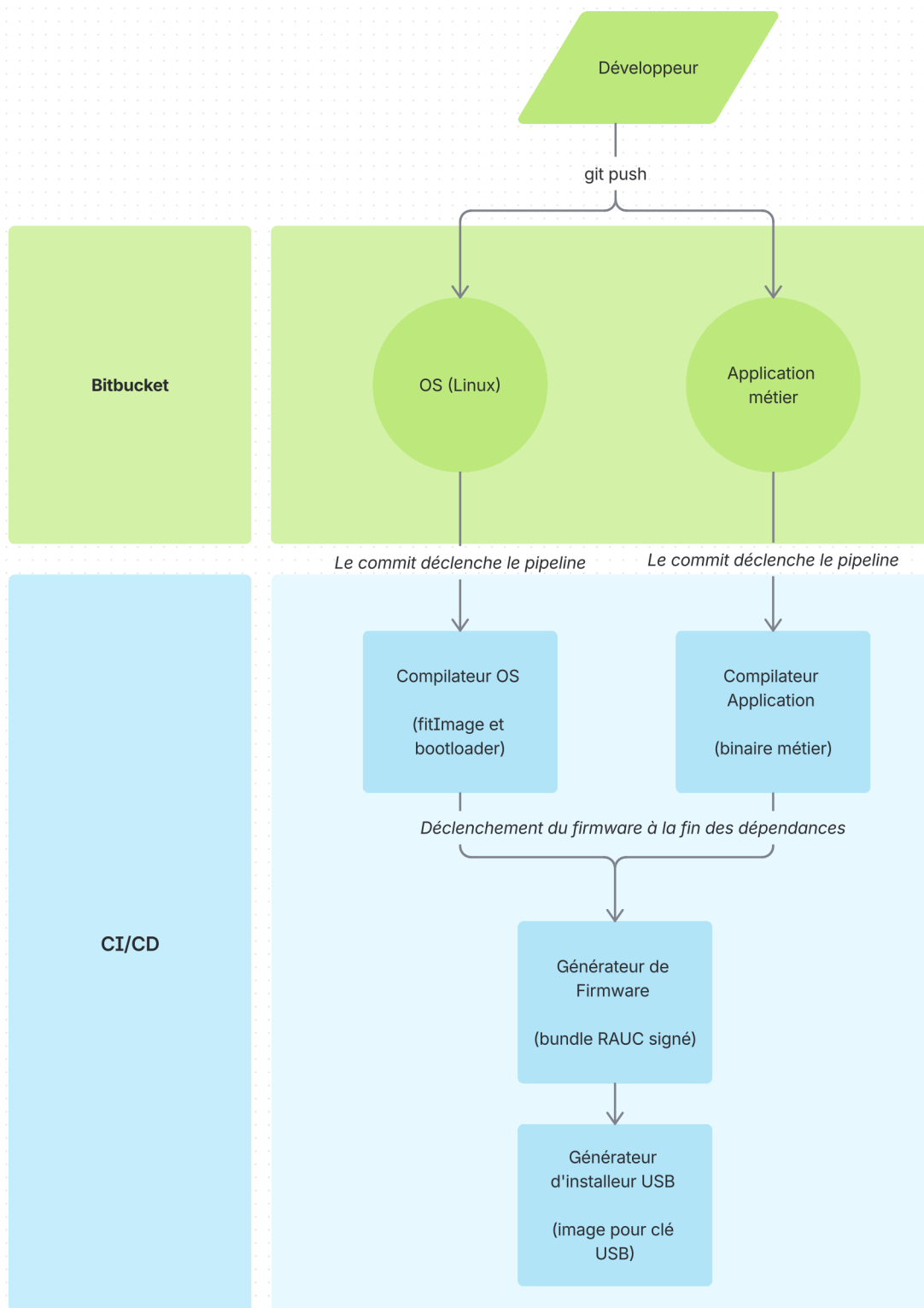


Figure 3.5.1 – Architecture des pipelines d'intégration continue (CI/CD)

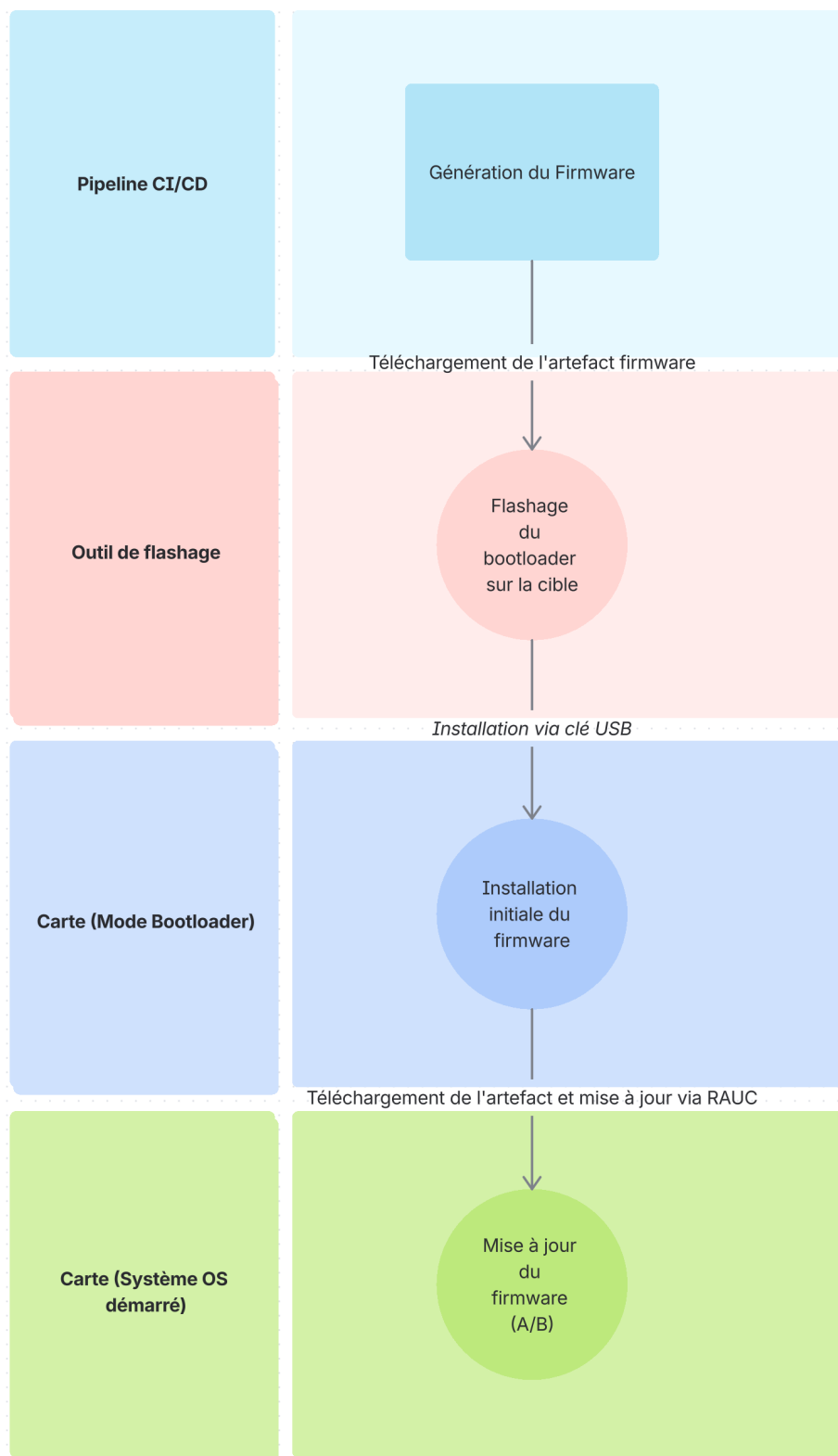


Figure 3.5.2 – Flux de déploiement et de mise à jour du firmware

4

Évaluation, prise de recul et impact de la mission

Ce chapitre présente une analyse critique des résultats obtenus au cours de la mission. Au-delà des validations techniques sur le système d'exploitation et la sécurité bas niveau, cette section met en lumière la prise de recul nécessaire à un travail d'ingénieur en évaluant les retombées de mes travaux sous quatre angles majeurs : technique, environnemental, organisationnel et stratégique/économique.

4.1 VALIDATION TECHNIQUE ET PERFORMANCES DE LA PLATEFORME

4.1.1 PROTOCOLE DE TEST ET PÉRIMÈTRE D'ÉVALUATION

La validation du système d'exploitation bas niveau et des mécanismes de sécurité s'est concentrée sur la couche OS, la chaîne d'amorçage et l'intégrité logicielle. Les bancs de

tests vidéo applicatifs pur étant pris en charge séparément par les équipes de validation produit, mon évaluation s'est portée sur trois indicateurs clés :

1. La vitesse et l'efficacité de la chaîne de boot (*Boot Time*).
2. L'optimisation du chargement de l'image noyau en mémoire RAM.
3. La robustesse du contrôle cryptographique des signatures.

4.1.2 OPTIMISATION DE L'AMORÇAGE ET RÉOLUTION DU GOULOT D'ÉTRANGLEMENT U-BOOT

Initialement, l'architecture x86 s'appuyait sur le *Slimbootloader* pour charger directement l'image FIT en mémoire RAM, en raison d'un problème d'extrême lenteur de chargement observé sous U-Boot. Cette configuration hybride introduisait une complexité inutile dans la chaîne d'amorçage et augmentait le temps de démarrage global.

En analysant et en résolvant ce dysfonctionnement au sein de la couche U-Boot, nous avons pu simplifier la séquence de boot en chargeant l'image FIT signée directement depuis U-Boot. Les résultats mesurés sont très satisfaisants :

- **Temps de démarrage global** : Le temps d'amorçage complet du système (du passage sous tension jusqu'à la disponibilité opérationnelle du système d'exploitation Linux) se situe désormais autour de 15 secondes. Ce temps d'exécution reste quasiment identique à celui mesuré avant l'intégration du *Secure Boot*, ce qui représente une réussite notable compte tenu de la surcharge de calcul (*overhead*) induite par la vérification cryptographique des signatures FIT et du déballage en mémoire.
- **Simplification architecturale** : Élimination de la dépendance complexe au chargement direct par le Slimbootloader, garantissant une meilleure maintenabilité du code source sans dégradation des performances temporelles.

4.1.3 VALIDATION DE LA CHAÎNE DE CONFIANCE CRYPTOGRAPHIQUE

Le processus de signature des images FIT utilise des paquets de clés RSA générés via OpenSSL. Lors de la séquence de boot, les fonctions natives d'U-Boot dédiées à la vérification de signature ont été intégrées et testées avec succès. Chaque tentative d'amorçage avec un binaire dont le hachage ou la signature RSA a été altéré provoque une

interruption immédiate du bootloader, confirmant l'imperméabilité du système face aux injections de code non autorisé.

4.2 IMPACT SUR L'OPTIMISATION ÉNERGÉTIQUE

4.2.1 SOBRIÉTÉ LOGICIELLE ET RÉDUCTION DE L'EMPREINTE OS

L'impact environnemental des technologies du numérique réside de plus en plus dans la consommation des équipements déployés en continu sur le terrain. L'approche retenue lors de la configuration de nos OS sous Yocto et Buildroot s'inscrit dans une démarche d'éco-conception logicielle :

- **Élimination des composants superflus** : Suppression systématique des démons, bibliothèques et paquets inutilisés à l'application métier vidéo, réduisant ainsi la taille du système de fichiers (*RootFS*) au strict minimum.
- **Réduction de l'activité CPU à l'amorçage** : L'optimisation du temps de boot diminue la phase de charge maximale du processeur lors des démarrages et redémarrages des équipements.

4.2.2 EFFICIENCE THERMIQUE ET PROLONGATION DE LA DURÉE DE VIE MATÉRIEL

Sur les plateformes nomades et compactes telles que le Falkon FX ou le module CellLink5G, la moindre sollicitation inutile du processeur génère de la chaleur résiduelle. En optimisant les drivers bas niveau et la gestion de la mémoire, mon travail contribue à maintenir les processeurs dans des plages de température optimales. Cela réduit l'usure thermique des composants électroniques embarqués et prolonge significativement leur durée de vie opérationnelle en conditions d'extrême mobilité.

4.3 OPTIMISATION DES FLUX DE TRAVAIL ET AGILITÉ DE L'ÉQUIPE

4.3.1 TRANSFORMATION DU TRAVAIL QUOTIDIEN DES DÉVELOPPEURS R&D

Avant la mise en place de la chaîne d'intégration continue et du système de packaging RAUC, la génération d'images pour les tests nécessitait des compilations manuelles locales complexes et chronophages.

L'automatisation via les pipelines sous Azure DevOps et TeamCity a profondément modifié les habitudes de l'équipe R&D :

- **Autonomie et rapidité** : Les développeurs de l'équipe applicative peuvent désormais pousser leur code sur Bitbucket et récupérer automatiquement des bundles RAUC et des images d'installation WIC dûment signés, sans avoir à maîtriser les subtilités des builds Yocto/Buildroot.
- **Décohabitation dev/prod** : La séparation claire entre le SDK de développement rapide (pour compiler des briques isolées) et les recettes de production (garantissant la reproductibilité) a permis de supprimer les frictions entre les phases d'exploration et de livraison.

4.3.2 REPRODUCTIBILITÉ ET TRAÇABILITÉ DES FIRMWARES

Grâce à la centralisation du suivi sur Kanban et au versionnage strict sous Git/Bitbucket, chaque binaire généré est directement traçable jusqu'au *commit* d'origine. Cette structuration améliore considérablement la synergie entre l'équipe *Embedded Systems*, l'équipe matérielle (FPGA/Électronique) et l'équipe Assurance Qualité (QA).

4.4 DE LA R&D À L'INDUSTRIALISATION : ROI ET AVANTAGE CONCURRENTIEL

4.4.1 ANALYSE DU RETOUR SUR INVESTISSEMENT (ROI)

En matière de coûts opérationnels pour un constructeur d'équipements vidéo professionnels comme Haivision, le retour sous garantie et les interventions de maintenance sur site représentent un poste budgétaire critique.

L'intégration de la solution RAUC associée à une architecture à double partition avec environnement de secours (*rescue*) apporte un gain économique direct :

- **Élimination du risque de panne définitive (*Bricking*)** : En cas de coupure de courant pendant un rafraîchissement système ou d'échec de démarrage d'un binaire, le mécanisme d'A/B fallback bascule automatiquement sur la version stable précédente ou sur la partition rescue.
- **Réduction drastique des coûts de Service Après-Vente (SAV)** : Les opérations de maintenance corrective peuvent s'effectuer à distance de manière totalement

sécurisée, évitant le retour coûteux des boîtiers en atelier pour reflashage physique.

4.4.2 AVANTAGE CONCURRENTIEL ET COMMERCIALISATION

Dans les secteurs hautement exigeants de la défense, de la sécurité publique (marchés ISR) et du broadcast critique, la sécurité logicielle est devenue un critère de choix discriminant.

- **Valorisation de l'offre commerciale** : Proposer des plateformes intégrant nativement un démarrage sécurisé certifié (*Secure Boot*) avec racine de confiance matérielle (*Root of Trust*) permet à Haivision de se positionner en tête lors d'appels d'offres stratégiques face à des concurrents ne disposant pas d'un tel niveau de durcissement.
- **Anticipation des normes européennes** : En garantissant le *Security by Design* et en intégrant un système automatisé de mise à jour sécurisée, mes travaux garantissent la conformité anticipée de la nouvelle gamme d'équipements (Makito ONE, Falkon FX, CellLink5G) avec le nouveau cadre réglementaire européen Cyber Resilience Act (CRA). Cette anticipation évite à l'entreprise tout risque de blocage de commercialisation sur le marché européen à l'échéance de la directive.

5

Conclusion et perspectives

5.1 BILAN TECHNIQUE ET SCIENTIFIQUE

Ce travail de fin d'études au sein de l'équipe *Embedded Systems* d'Haivision marque l'aboutissement d'un cycle complet d'ingénierie, faisant évoluer des architectures d'exploration R&D vers des solutions logicielles bas niveau industrialisables et prêtes pour la production.

L'ensemble des objectifs techniques et opérationnels fixés au début de la mission ont été atteints avec succès. Les livrables majeurs transmis à l'entreprise se résument ainsi :

- **Validation et intégration du code source** : La branche de développement regroupant l'implémentation du démarrage sécurisé (*Secure Boot* avec vérification d'images FIT signées RSA) et le système de mise à jour redondant (*RAUC*) a été entièrement validée, revue et fusionnée (*merged*) dans la branche principale (*main*) des dépôts d'Haivision.

- **Opérationnalité des chaînes d'intégration continue** : Les pipelines automatisés sous Azure DevOps (ADO) et TeamCity sont désormais totalement fonctionnels. Ils assurent la compilation croisée, la signature cryptographique et la génération automatique des bundles de firmware (.raucb) et des images d'installation initiale pour les différentes plateformes.
- **Architecture de stockage sécurisée** : Le schéma de partitionnement standardisé et la protection du système de fichiers ont été déployés sur les architectures ARM et x86.

Ces réalisations garantissent la stabilité, la traçabilité et l'intégrité logicielle de la nouvelle génération d'équipements de transmission vidéo d'Haivision.

5.2 PERSPECTIVES INDUSTRIELLES ET TECHNIQUES

Bien que les jalons principaux de cette mission soient validés, plusieurs axes d'évolution techniques s'inscrivent dans la continuité des travaux menés pour Haivision :

1. **Certification de sécurité et conformité au Cyber Resilience Act (CRA)** : La prochaine étape consistera à poursuivre les audits de sécurité internes et externes afin d'identifier d'éventuelles failles ou vulnérabilités résiduelles. Ce processus de durcissement continu permettra d'obtenir les certifications de sécurité officielles requises pour garantir la pleine conformité des produits avec la réglementation européenne imminente (CRA).
2. **Support et évolution de la couche matérielle-logicielle** : L'équipe *Embedded Systems* continuera d'assurer le support transversal pour l'ensemble des projets de l'entreprise, notamment à travers la maintenance des distributions Linux (Yocto/Buildroot), le développement de drivers de périphériques spécifiques et l'optimisation continue des interactions matériel/logiciel.

5.3 BILAN PERSONNEL ET MONTÉE EN COMPÉTENCES

Sur le plan personnel et professionnel, cette expérience au sein d'Haivision constitue une étape décisive dans la construction de mon profil d'ingénieur. Ce parcours m'a permis d'appréhender le cycle de vie complet d'un produit électronique professionnel : depuis la

phase d'exploration technologique initiale lors de mon stage précédent, jusqu'à la compréhension intime des plateformes déjà déployées et l'évolution du code vers une nouvelle architecture industrielle.

Au-delà des compétences purement techniques en développement Linux bas niveau, cette mission a fortement développé mes aptitudes organisationnelles et managériales :

- **Prise d'initiative et leadership technique** : La nécessité d'orienter les choix d'architecture (partitionnement, RAUC, clés cryptographiques) m'a amené à planifier et animer des réunions de cadrage technique pluridisciplinaires, renforçant ma capacité à convaincre, synthétiser et trancher en contexte professionnel.
- **Complémentarité école / entreprise** : Cette mission a offert un terrain d'application directe idéal aux connaissances théoriques et pratiques acquises lors de ma dernière année d'études. Les compétences en réseaux de transmission issues de la TAF STAR et l'expertise en architectures embarquées développée dans la TAF SEH d'IMT Atlantique — combinées aux bases en télécommunications de mon cursus à l'UDELAR — ont constitué un atout majeur pour réussir cette mission.

En conclusion, ce contrat de professionnalisation complète ma formation académique en matérialisant mon projet professionnel : devenir un ingénieur reconnu dans le domaine des systèmes embarqués, capable d'allier rigueur scientifique, maîtrise logicielle bas niveau et vision stratégique industrielle.

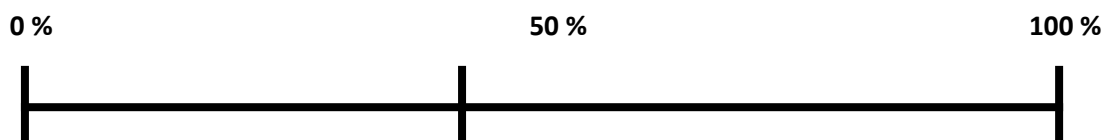
Bibliographie

- [1] ARM Limited. *ARM Trusted Firmware-A (TF-A) Architecture and Firmware Image Package (FIP)*, 2024. URL <https://trustedfirmware-a.readthedocs.io/>.
- [2] Buildroot Developers. *The Buildroot User Manual*, 2025. URL <https://buildroot.org/downloads/manual/manual.html>. Consulté en 2026.
- [3] Das U-Boot Project. *U-Boot Documentation : Flattened Image Tree (FIT) Verification*, 2024. URL <https://u-boot.readthedocs.io/en/latest/usage/fit/index.html>.
- [4] European Parliament and Council of the European Union. Regulation (eu) 2024/2847 of the european parliament and of the council on horizontal cybersecurity requirements for products with digital elements (cyber resilience act). *Official Journal of the European Union*, 2024. URL <https://eur-lex.europa.eu/eli/reg/2024/2847/oj>.
- [5] Haivision Systems Inc. Safe stream transport (sst) technology and emmy® award recognition, 2024. Technologie propriétaire d'agrégation de liaisons cellulaires.
- [6] Haivision Systems Inc. *Makito ONE Video Transport Platform Datasheet*, 2025. Document officiel Haivision.
- [7] NXP Semiconductors. *i.MX 95 Applications Processor Family Reference Manual*, 2024. Architecture ARM Cortex-A55 / Cortex-M33 / NPU.
- [8] Pengutronix. *RAUC - Robust Auto-Update Controller Documentation*, 2025. URL <https://rauc.readthedocs.io/>. Version 1.11+.
- [9] SRT Alliance. Srt protocol technical specification. Technical report, Haivision Systems Inc., 2020. URL https://github.com/Haivision/srt/blob/master/docs/SRT_Protocol_Technical_Specification.md.
- [10] The Yocto Project. *The Yocto Project Documentation*, 2025. URL <https://docs.yoctoproject.org/>. Consulté en 2026.

Annexe - Utilisation de l'IA

1. Si vous avez utilisé l'IA pour la rédaction de votre rapport,

Estimez la proportion de votre travail qui a été impactée par l'utilisation de l'IA.



Cochez les cases correspondant à votre utilisation de l'IA pour ce travail :

EXPLORATION ☐ S'inspirer ☐ Générer des idées ☐ Approfondir un sujet ☐ Recherche documentaire ☐ Autre : (précisez) ...	AMÉLIORATION ☒ Corriger ses fautes ☒ Reformuler ses phrases ☐ Demander des conseils ☐ Autre : (précisez) ...
RÉSOLUTION DE TÂCHES ☐ Synthétiser ☐ Répondre à une question ☐ Autre : (précisez) ...	PRODUCTION DE CONTENUS ☒ Générer un plan ☒ Rédiger des parties d'un texte ☐ Générer des infographies ☐ Autre : (précisez) ...

2. Si vous n'avez pas utilisé l'IA pour la rédaction de votre rapport, qu'est-ce qui a motivé votre décision ?

(exemple : votre entreprise vous l'interdit ? Vous avez une opinion négative des IA ? Vous avez préféré mobiliser vos propres compétences ?)

--