



Participation à l'innovation d'un système-sur-puce dans une start-up de la Silicon Valley

Rapport « court » de stage Jeune Ingénieur

27 juillet 2007 - 25 juillet 2008

Grégoire Péan

gregoire.pean@telecom-bretagne.eu

ENTREPRISE

Mobilygen Corporation

Santa Clara, Californie

ENCADRANTS CHEZ MOBILYGEN

Rémi Machet | Juillet 2007 - Mars 2008

Software Manager

rmachet@mobilygen.com

Chris Wein | Mars 2008 - Juillet 2008

Director of Software

cwein@mobilygen.com

ENCADRANT ÉCOLE

Jean-Marie Gilliot

jm.gilliot@telecom-bretagne.eu

Table des matières

Table des matières	2
1 Introduction	3
2 Contexte du stage	5
2.1 Mobylygen Corp., une <i>start-up</i> de la Silicon Valley	5
2.2 Environnement, organisation du travail et management « à l'américaine »	6
3 Problématique du stage	9
3.1 Développement pour le système-sur-puce	9
3.2 Méthodologie	10
3.3 Gestion du projet dans une logique de résultats	10
4 Détails des lots technologiques (<i>work packages</i>)	13
4.1 Conception d'un sous-système d'entrées-sorties généralistes (GPIO)	13
4.2 Optimisation des performances du <i>streaming</i> avec le protocole UDP	14
4.3 Utilisation de l'interface SPI pour la réception d'un flux vidéo	16
4.4 Résultats globaux	16
5 Conclusion	17
Bibliographie	19

Introduction

Ce stage d'un an, réalisé à l'issue de ma deuxième année à TELECOM Bretagne, au sein de Mobylygen Corp., a été pour moi une occasion formidable de démarrer en entreprise mon futur métier d'ingénieur. J'ai validé ce stage en septembre dernier comme stage de fin d'étude. J'ai eu la chance d'assister et/ou prendre part aux différentes étapes de la création d'une puce complexe, depuis sa conception jusqu'à l'assistance aux clients. J'ai été impliqué dans la vie de l'entreprise, au-delà de mes attentes. Après les six premiers mois du stage, j'étais un membre de l'entreprise à part entière, complètement intégré aux équipes ; on m'accordait une grande confiance, m'offrant l'opportunité de prendre des responsabilités et me laissant prendre des initiatives personnelles. Ce stage m'a donc apporté de grandes satisfactions, les missions confiées et le travail accompli ayant été particulièrement valorisants.

Durant ce stage, j'ai principalement géré un projet de développement logiciel pour la puce phare de Mobylygen, une puce de codage/décodage audio-vidéo, avec des contraintes/délais/innovation. J'avais déjà une expérience du développement logiciel pour des projets logiciels intermédiaires en tant que consultant *freelance*, mais c'était toujours à un niveau « utilisateur final », applicatif, avec peu de contraintes.

Dans le contexte de Mobylygen, c'était différent : j'ai touché aux couches logicielles très basses, s'interfaçant parfois directement avec le matériel. Cela soulève plus de problèmes car c'est un environnement complexe et délicat. Il faut faire preuve d'une très grande rigueur car la moindre erreur dans la manière dont vous interagissez avec ce matériel peut être fatale à la stabilité du produit.

J'ai également rencontré des contraintes d'études avec des objectifs serrés vis-à-vis des clients : une vente peut dépendre de peu de choses, en particulier quand le produit est toujours en création. J'étais en lien direct avec l'équipe *marketing* et parfois les clients, pour corriger des bogues ou apporter des fonctionnalités qui étaient la condition pour remporter un marché. Apporter des solutions à des problèmes complexes dans un environnement contraint, ce fut la « substance » de ce stage.

La phase de conception n'était pas absente non plus. Créer des solutions logicielles pérennes implique de ne pas rater la conception dudit logiciel – l'entreprise préfère des solutions qui n'ont pas à être réécrites chaque fois qu'un client introduit une nouvelle exigence, ou qu'une nouvelle puce est créée. Par exemple, j'ai dû créer des pilotes de périphériques articulés en plusieurs couches pour maximiser leur portabilité parmi les différentes puces de Mobylygen.

Dans un premier chapitre, je ferai une présentation de l'entreprise et du contexte du stage. Dans un second chapitre, je présenterai la problématique du stage, puis poursuivrai par une présentation en détail des lots technologiques. En conclusion, j'indiquerai l'intérêt des résultats obtenus et ce que j'en ai retiré comme apports personnels.

Contexte du stage

2.1 Mobylygen Corp., une *start-up* de la Silicon Valley

Mobylygen est une *start-up* située en plein coeur de la *Silicon Valley*. Elle a été créée en 2000 par Sorin Cismas et a commencé le développement de son premier produit en 2003. Elle compte aujourd'hui environ 80 employés. Mobylygen conçoit des puces spécialisées d'encodage et décodage audio-vidéo, au format H.264 AVC, une norme de compression d'une grande efficacité. Ces puces trouvent leur place dans différents appareils électroniques, comme des caméscopes, caméras de surveillance et autres *webcams*, et plus généralement partout où il est intéressant de transférer et/ou stocker un flux vidéo en minimisant le débit et l'espace utilisé tout en conservant une qualité d'image satisfaisante. Mobylygen conçoit mais ne fabrique pas à proprement parler les puces : on parle d'entreprise de semiconducteurs *fabless* (sans fabrication). La fabrication étant sous-traitée en Asie. L'entreprise ne vend pas de produits basés sur ses puces à l'utilisateur final, mais propose cependant des *reference designs* (modèles de produits typiques basés sur les puces) à ses clients qui se chargent de faire des produits finis pour le grand public. Elle crée de la « valeur » par l'innovation.

Mobylygen a déjà à son actif 38 brevets et s'est imposée dès le début comme une entreprise à la pointe dans son secteur en annonçant la première puce d'encodage/décodage H.264 au monde en 2004.

Mobylygen, c'est à ce jour trois puces : deux en développement et une finalisée. Parmi celles en développement, il y a le MG3500, puce sur laquelle j'ai travaillé pendant mon stage. Quand je suis arrivé en juillet 2007, le *design* du MG3500 venait tout juste d'être envoyé au fondeur taïwanais. J'ai donc pu suivre et participer à toutes les étapes qui ont suivi la réception des premiers exemplaires.

Plusieurs équipes travaillent ensemble sur les puces et chacune possède un rôle particulier : de la conception, au niveau « portes logiques », au développement des logiciels, en passant par le support des clients et le *marketing*, l'entreprise regroupe des pôles de compétences de pointe et d'excellence variés au sein de sa petite structure.

Pour ma part, **j'ai travaillé dans l'équipe SoC (pour « System-on-Chip »)**. Cette équipe, composée de six personnes, a la responsabilité de fournir les logiciels qui permettent à la puce de communiquer avec le monde extérieur. Cela signifie développer la partie logicielle qui est exécutée par la puce et qui lui permet d'utiliser de manière autonome ses interfaces de communication (comme USB, Ethernet, etc.). Cela signifie aussi s'occuper des logiciels qui prennent

place sur les puces extérieures quand le MG3500 fonctionne en mode co-processeur (en tant qu’esclave spécialisé d’une autre puce). C’est aussi l’équipe qui effectue le plus gros du travail pendant la phase dite de *bring-up*, qui est la phase des tests quand les premiers exemplaires reviennent de l’usine.

Cette équipe est en permanence à la limite entre logiciel et matériel : il faut régulièrement utiliser les oscilloscopes et l’analyseur logique pour résoudre un problème. Il faut souvent se référer aux *datasheets* des puces tierce-partie, documentation de protocoles divers, etc. Le code que vous écrivez est toujours en lien étroit avec le matériel : une grosse partie de ce code consiste en des pilotes (pilotes Linux) pour les interfaces matérielles de la puce.

De par sa position à l’interface entre logiciel et matériel, l’équipe SoC a un rôle central dans l’entreprise. Cela permet de cotoyer beaucoup de personnes différentes et de s’attaquer à des problèmes hétéroclites.

2.2 Environnement, organisation du travail et management « à l’américaine »

À mon arrivée à Mobilygen, les premiers exemplaires de la puce MG3500 revenaient du fondeur, et la phase dite de *bring-up* commençait : les équipes SoC et *Software* avaient besoin d’utiliser les plateformes de développement intensivement.

Les plateformes de développement étaient au départ en nombre restreint (5 plateformes pour une vingtaine d’employés). Par la suite, à ma propre initiative, comme il était crucial que les employés soient en mesure de savoir quelles plateformes étaient utilisées à un instant donné, j’ai créé un script de supervision de l’utilisation des plateformes, accessible depuis le navigateur web de tous les employés. Ce script était en interaction permanente avec les plateformes de développement et, en plus de maximiser l’utilisation de celles-ci et la productivité des employés, il me permettait de surveiller les problèmes qui surgissaient occasionnellement à la suite d’une mise à jour de la distribution Linux par moi-même.

Board	Locked By	Idle Time	Other Info
merlinbup1 10.10.10.223 Comments: Feng's office		--:--	Has Merlin daughter board: Vcore=994mV Connections: 0 (none) Avg. Load: 0.00, 0.00, 0.00 Uptime: 5:34:-- CF Usage: 444M / 473M MIPS Dist: Mobilygen Linux distribution glibc2.6.1.ver1.pl9 MIPS Kernel: 2.6.22.mobi.mips-devkit-merlinbup.r1431
merlinbup2 10.10.10.182 Comments: none		--:--	Has Merlin daughter board: Vcore=1004mV Connections: 0 (none) Avg. Load: 0.00, 0.00, 0.00 Uptime: 7 days CF Usage: 441M / 472M MIPS Dist: Mobilygen Linux distribution glibc2.6.1.ver1.pl9 MIPS Kernel: 2.6.22.mobi.mips-devkit-merlinbup.r1431
merlinbup12 10.10.10.181 Comments: Anthony's Cubicle	athai	For user athai: 0:04:47	Has Merlin daughter board: Vcore=1004mV Connections: 2 (athai) Avg. Load: 0.03, 0.13, 0.28 Uptime: 1 day CF Usage: 441M / 481M MIPS Dist: Mobilygen Linux distribution glibc2.6.1.ver1.pl9 MIPS Kernel: 2.6.22.mobi.mips-devkit-merlinbup.r1431
merlinbup13 10.10.10.111 Comments: Ian's cube	cwein	For user cwein: 6:06:00	Has Merlin daughter board: Vcore=1055mV Connections: 1 (cwein) Avg. Load: 0.00, 0.00, 0.00 Uptime: 97 days CF Usage: 431M / 481M MIPS Dist: Mobilygen Linux distribution glibc2.6.1.ver1.pl9 MIPS Kernel: 2.6.22.mobi.mips-devkit-merlinbup.r1431
merlinbup14 Comments: in the lab		---	--- ONLINE but unable to retrieve info, try again later ---
merlinbup15 Comments: none			--- OFFLINE ---
merlinbup16			OFFLINE

FIGURE 2.1: Capture d’écran de la page du script de supervision

Travailler dans la Silicon Valley, et notamment dans une petite structure, comporte quelques particularités intéressantes. Par exemple, la hiérarchie est très effacée et il n’y a pas le sentiment qu’un employé a la solution ultime parce que son poste est hiérarchiquement plus haut placé. Ainsi, quand une équipe rencontre un problème, tous les avis sont bons à prendre et tous les avis sont pris, et il n’est pas mal vu d’argumenter avec son supérieur. Il n’est pas mal vu de venir proposer spontanément une idée à n’importe qui, pas nécessairement dans la même équipe. En fait, les échanges sont très encouragés, quels qu’ils soient, car ils font dans tous les cas avancer la réflexion, d’un côté ou de l’autre. Une large place est également accordée à la communication par courriel.

Un exemple notable de cette priorité des échanges sur la hiérarchie est le fait que pendant mon stage, **j’ai moi-même auditionné cinq personnes** candidates pour rejoindre l’équipe SoC. L’une postulait pour remplacer mon ancien tuteur et supérieur à la tête de l’équipe, Rémi Machet. Les autres candidaient pour un poste d’ingénieur senior. Encore une fois, cela est très valorisant pour un stagiaire – et on s’efforce d’être digne de la grande confiance qui vous est accordée.

Il est assez intéressant de noter que le travail était très orienté « **résultats** » et « **échéances** ». Ainsi, la correction de bogue critique pour remporter un contrat **m’a permis de toucher un bonus pécunier**. A l’inverse, la non-réalisation d’une tâche dans les délais impartis provoquait parfois la réorganisation des ressources pour vous aider dans cette tâche (un autre employé venait vous appuyer).

Plus largement, une analyse SWOT permet de bien cerner les points positifs et négatifs liés au travail dans une *start-up*.

<p><i>Forces :</i></p> <ul style="list-style-type: none"> • Réactivité, dynamisme, flexibilité • Prises de décisions rapides • Implication individuelle importante : le travail de chacun est visible • Intéressement des salariés et prise de participation dans l’entreprise • Autonomie • Management participatif 	<p><i>Faiblesses :</i></p> <ul style="list-style-type: none"> • Vision à court terme : précarité entraîne recherche du succès immédiat • Tutorat, encadrement peu important • L’information est souvent détenue par une seule personne (« l’expert ») • Positionnement sur des niches • Dénomination de l’entreprise peu connue • Difficulté à répondre à des gros appels d’offre • Crédibilité face à des gros clients
<p><i>Opportunités :</i></p> <ul style="list-style-type: none"> • Innovation pure, création de valeur • Localisation dans la <i>Silicon Valley</i> : émulsion, catalyse due à la concentration d’entreprises et de cerveaux • Fusion/acquisitions par des groupes plus importants • Prise de risque facilitée par le mode de management des fondateurs entrepreneurs dans l’âme • Capacité à valider des normes internationales (CCMI) 	<p><i>Menaces :</i></p> <ul style="list-style-type: none"> • Manque de « cash » (financement externe) • Manque de ressources humaines • Concurrence féroce • Manque d’information sur les brevets et la valorisation • Espionnage industriel et piratage

FIGURE 2.2: Analyse SWOT de la *start-up* dans la *Silicon Valley*

Problématique du stage

3.1 Développement pour le système-sur-puce

En fait, *System-on-Chip*, le nom de l'équipe dans laquelle j'ai travaillé, signifie « système-sur-puce ». Un système-sur-puce est **un système électronique et informatique autonome** intégré sur une même puce : l'augmentation de la densité des transistors permet de nos jours d'intégrer **sur un même morceau de silicium** un microprocesseur, un contrôleur mémoire, des contrôleurs et interfaces permettant le dialogue avec l'environnement comme le ferait un PC de bureau (par exemple, interfaces USB, SATA, Ethernet).

Le MG3500 possède donc un système-sur-puce, et mes missions principales ont consisté à **concevoir, déboguer, optimiser ou valider des pilotes de périphérique Linux et des solutions, dans les délais et coûts impartis** pour ce système-sur-puce. En effet, une distribution Linux est exécutée sur le système-sur-puce, et des pilotes sont nécessaires pour pouvoir gérer les interfaces (USB, Ethernet, etc.) depuis Linux.

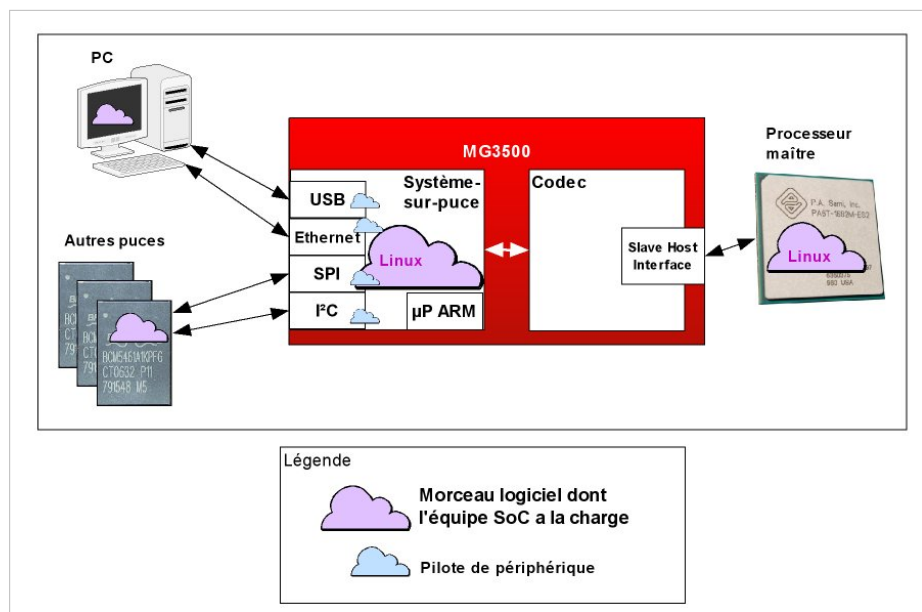


FIGURE 3.1: Schéma simplifié des interactions du MG3500 avec son environnement, qui montre le travail effectué par l'équipe SoC au niveau logiciel

3.2 Méthodologie

L'écriture de pilotes de périphérique pour le noyau Linux est une tâche rigoureuse : les ressources disponibles sont très limitées, et bien définies. Une erreur peut provoquer une instabilité du système complet. Il existe deux ouvrages de référence complémentaires : « *Understanding the Linux Kernel*[1] » de Daniel Bovet et Marco Cesati, qui propose une explication du fonctionnement du noyau dans son ensemble, et « *Linux Device Drivers*[3] », Jonathan Corbet et Alessandro Rubini, de qui fournit une aide pour l'écriture des pilotes eux-mêmes.

On rencontre plus précisément des problèmes de *réentrance* et d'*interruptions*, de *contexte d'exécution* qui sont autant de choses à prendre en compte et qui sont particuliers à la programmation système.

Pour le débogage, les outils sont malheureusement très limités, et il faut souvent se contenter d'imprimer des choses à la console pour **confirmer ou infirmer des hypothèses**. C'était la seule méthodologie souvent viable pour résoudre des problèmes : **l'émission d'hypothèses les plus étroites possible et l'expérimentation** pour les valider ou les invalider.

Une grande part d'intuition (ou plutôt, d'expérience) est nécessaire pour faire le plus rapidement possible les bonnes hypothèses ; ce qui est crucial car toute hypothèse prend en moyenne une dizaine de minute à valider (recompilation du pilote, voire du noyau complet). Cette approche expérimentale est contraignante, mais cela vous oblige à **bien comprendre ce que fait votre code**, et vous fait progresser.

Pour chaque pilote développé, je me suis efforcé de suivre les lignes de conduite suivantes pour assurer la pérennité de mon travail dans l'entreprise, faciliter son amélioration, voire permettre sa soumission à la communauté du noyau Linux :

- respect des conventions de mise en forme du code et de codage du noyau Linux[2] ;
- code et commentaires intégralement en Anglais ; noms de variables et fonctions significatifs et ne collisionnant pas avec les autres pilotes et sous-systèmes du noyau ;
- indépendance des pilotes de l'architecture : un pilote doit pouvoir être utilisé sur une autre architecture que la puce de Mobilygen, quand cela est possible ;
- code optimisé pour la rapidité d'exécution et l'utilisation minimales des ressources ; par exemple par l'utilisation des fonctions `likely` et `unlikely`¹ qui donnent des indices au compilateur pour optimiser le code au mieux selon la probabilité d'apparence d'une condition ;
- code systématiquement réentrant pour garantir l'utilisation stable des pilotes même dans d'autres architectures (à processeurs ou cœurs multiples) ;
- *design* orienté objet quand cela est possible.

3.3 Gestion du projet dans une logique de résultats

Ci-dessous, un diagramme de répartition temporelle des différentes tâches effectuées permet de cerner la dynamique du stage et les échéances.

1. <http://kerneltrap.org/node/4705>

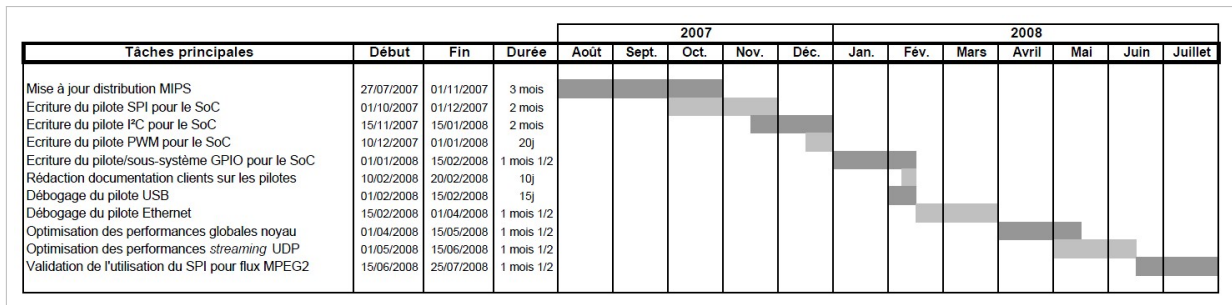


FIGURE 3.2: Diagramme de répartition temporelle des tâches

Le travail dans une petite entreprise implique une logique de résultats rapides pour un retour sur investissement. Des réunions d'équipe hebdomadaires mais également un système de *reporting* avec mon chef d'équipe m'ont permis de progresser de manière continue.

Ainsi, par exemple l'écriture du pilote SPI pour le SoC a été réduit à deux mois de travail au lieu de quatre mois prévus initialement. Le temps dégagé m'a permis de remplir des missions supplémentaires non prévues initialement : l'optimisation des performances et la validation ont ainsi pu être réalisées alors qu'elles n'avaient pas été prévu dans le programme initial.

J'ai apprécié la communication interne très interactive qui permet une adaptation permanente.

Le diagramme de répartition temporelle des tâches est un outil de pilotage mais également de communication partagée qui est fondamental en terme de gestion de projet et d'expérimentation.

Détails des lots technologiques (*work packages*)

4.1 Conception d'un sous-système d'entrées-sorties généralistes (GPIO)

Pour l'interface GPIO (*General Purpose Input Output*), j'ai conçu, en plus du pilote « habituel », un sous-système. Ce dernier se présente comme une couche d'abstraction logique entre le pilote qui dépend directement du matériel utilisé pour les GPIO (pilote contrôleur) et les pilotes qui utilisent les GPIO mais qui ne dépendent pas spécifiquement du matériel utilisé pour ces GPIO (pilotes clients).

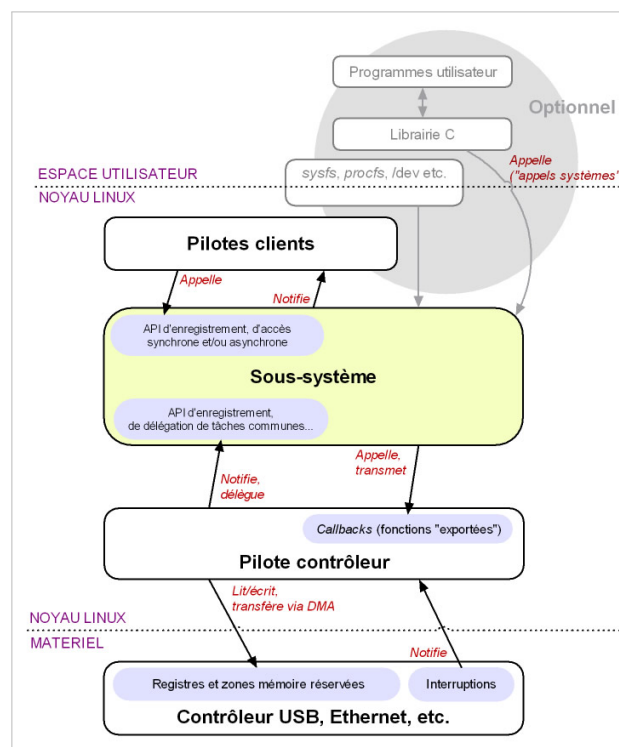


FIGURE 4.1: Situation d'un sous-système dans le noyau et interactions

Concevoir un sous-système implique de bien prévoir les mécanismes d'enregistrement et de notification de part et d'autre du sous-système. Il faut prévoir les risques d'accès concurrent à un même port GPIO, et fournir la plus grande flexibilité possible en même temps.

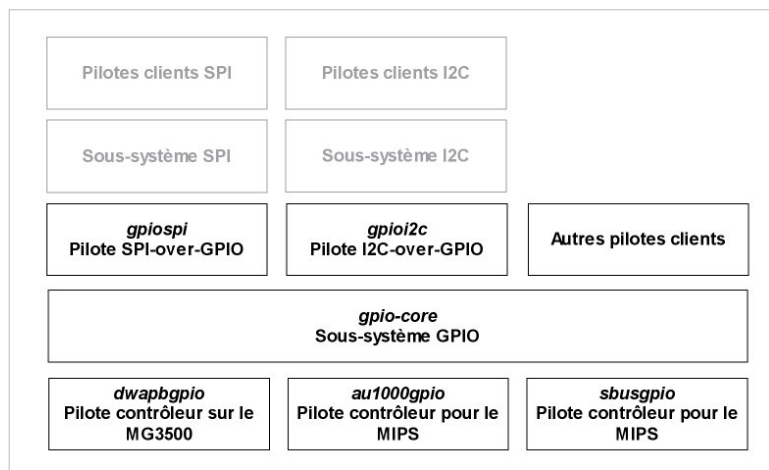


FIGURE 4.2: Le sous-système GPIO est au coeur d'une pile logicielle complexe

L'utilisation du sous-système GPIO maximise la portabilité des pilotes clients et minimise le travail à effectuer à chaque changement de matériel, à chaque nouvelle puce, **c'est donc une économie durable pour l'entreprise.**

4.2 Optimisation des performances du *streaming* avec le protocole UDP

Les performances n'étant pas à la hauteur pour le *streaming* UDP des flux vidéo, j'ai travaillé avec Stephan Lachowsky, ingénieur de l'équipe *Software*, pour aller plus loin. L'étude de la pile TCP/IP du noyau Linux et la lecture de publications[4] nous a amené à concevoir une architecture de transfert « zéro copie » depuis le codec jusqu'au pilote Ethernet, qui a permis de **quadrupler** les performances pour le *streaming* UDP.

Il a été question de breveter la solution trouvée, ce qui n'a, finalement, pas été fait.

Un schéma ci-dessous permet d'expliquer les solutions apportées et qui aujourd'hui donnent satisfaction.

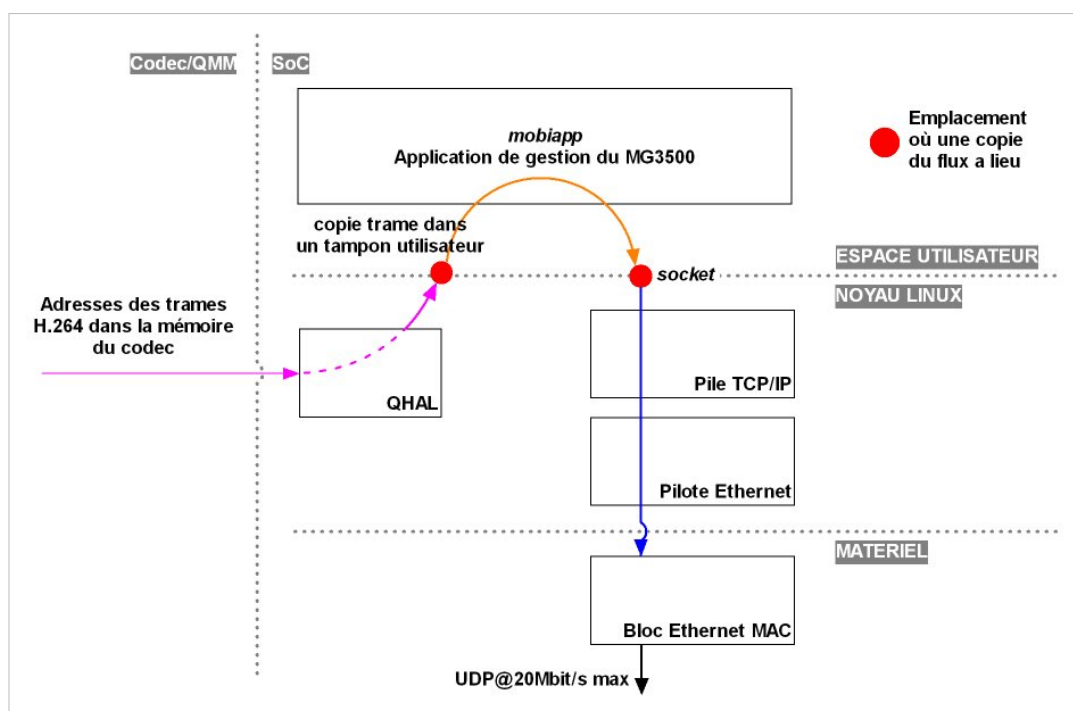


FIGURE 4.3: Schéma synoptique sans les modifications de zéro copie

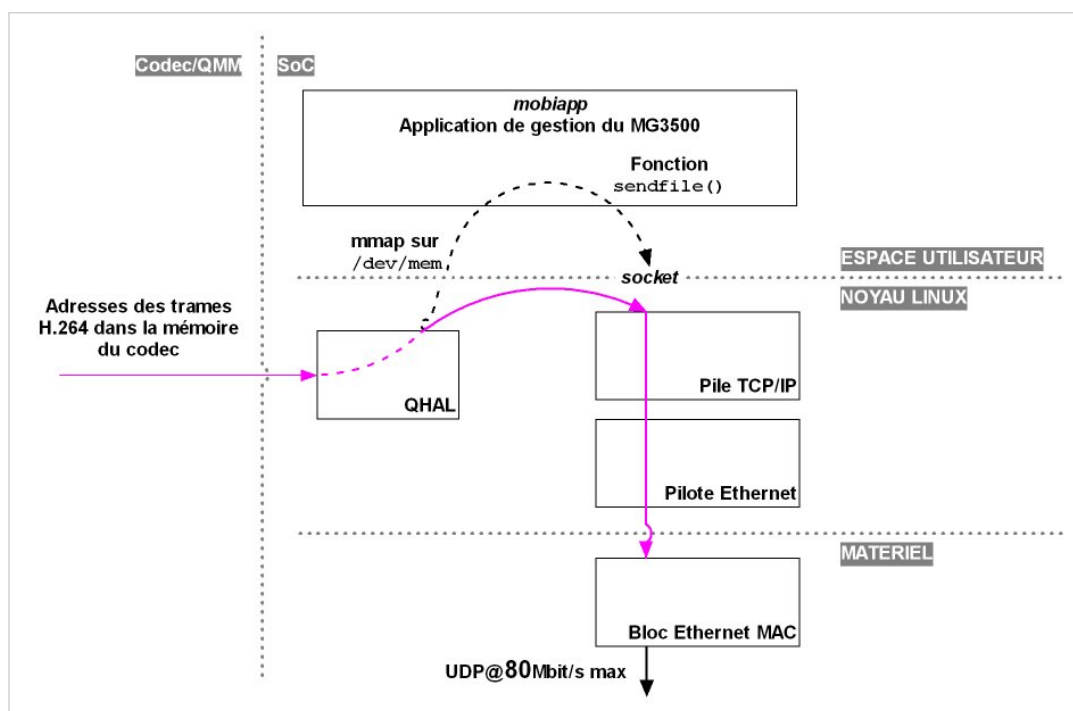


FIGURE 4.4: Schéma synoptique avec les modifications de zéro copie

4.3 Utilisation de l'interface SPI pour la réception d'un flux vidéo

Mon dernier lot technique a consisté à valider la transmission des données entre un circuit intégré *tuner* et l'interface SPI du MG3500. Si je démontrais qu'il était possible d'utiliser l'interface SPI pour transférer les données, cela permettait de se passer d'un circuit FPGA intermédiaire pour préconditionner le flux et utiliser une autre interface que SPI. Et un circuit intégré FPGA en moins, ce sont **des coûts de production et un encombrement réduits pour les clients**.

Le circuit *tuner* envoie le flux MPEG2 à 20Mbit/s via un protocole série non strictement compatible avec la spécification de la norme SPI de Motorola. Il a fallu vérifier que notre interface SPI sur le MG3500 supportait bien ce mode de transmission, et s'assurer aussi de l'intégrité des données à 20MHz, qui n'était pas garantie.

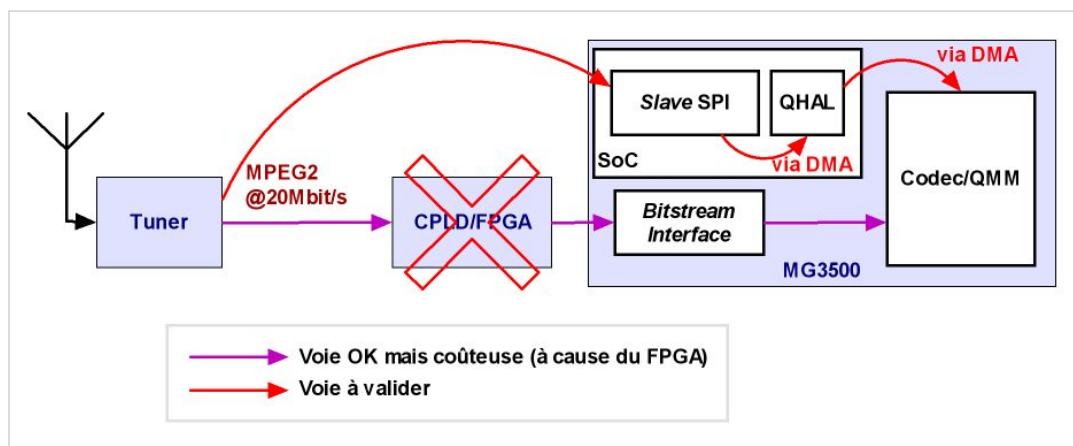


FIGURE 4.5: Schéma synoptique des deux cheminements possibles depuis le *tuner* jusqu'au MG3500

4.4 Résultats

L'intérêt de mon travail pour l'entreprise est que :

- tous les pilotes créés sont inclus au SDK fourni aux clients et cela fonctionne ;
- la documentation que j'ai rédigée est incluse au manuel fourni aux clients ;
- les optimisations apportées sont en place et ont permis de remporter des contrats supplémentaires ;
- la correction des bogues (USB, Ethernet) ont permis de crédibiliser le SDK auprès de clients réticents.

A titre anecdotique, ce stage peut se résumer en quelques chiffres :

- 321 *commits* au serveur SVN affectant 354 fichiers distincts ;
- 78252 lignes (de code, rustines, scripts...) ajoutées et 22869 lignes supprimées.

Conclusion

Ce stage m'aura permis d'améliorer ma perspicacité dans la résolution de problèmes complexes, faisant intervenir en même temps logiciel de bas niveau – programmation noyau – et matériel, dans un environnement hautement contraint : consommation électrique de la puce, ressources mémoire et de calcul disponibles, architectures bien spécifiques, échéances et pression imposées indirectement ou directement par les clients et la compétitivité qu'une petite entreprise se doit d'avoir pour survivre.

Du point de vue technique, j'ai su mettre à profit mon savoir existant, mais surtout apprendre les technologies nécessaires pour proposer des solutions viables et pérennes pour l'entreprise. J'ai pris des initiatives, pas seulement dans les contournements et solutions aux problèmes à trouver, mais aussi en mettant en place des dispositifs qui facilitent le travail des salariés (initiative de rédaction de pages de *How-To* dans le Wiki de l'entreprise) et améliorent la productivité (script de supervision des plateformes de développement).

Du point de vue humain, vivre aux Etats-Unis m'a permis de découvrir une culture finalement bien différente de la culture française et de m'ouvrir de manière plus importante aux autres, d'avoir moins d'a priori, et de perfectionner ma connaissance de la langue anglaise avec un léger accent californien !

Du point de vue gestion de projet, ce stage m'a permis de découvrir les outils de pilotage de lots techniques et de partager l'information. Le management à l'américaine, qui réussit bien à certaines entreprises de la *Silicon Valley* car la flexibilité apportée est adaptée à la réalité d'un marché dynamique et en constante évolution et innovation.

Je ne sais pas encore si je retournerai travailler en Californie (Mobilygen m'a d'ores et déjà fait une proposition d'embauche), mais si je décide de rester travailler en France ou dans un autre pays européen, cette expérience de 12 mois me permettra d'avoir un recul intéressant à l'international.

Bibliographie

- [1] Daniel Bovet, Marco Cesati. *Understanding the Linux Kernel, Third Edition*. O'Reilly Media, Inc., novembre 2005.
- [2] Greg Kroah-Hartman. *Proper Linux Kernel Coding Style*. Disponible sur <http://www.linuxjournal.com/article/5780> (consulté le 2 septembre 2008), 2002.
- [3] Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman. *Linux Device Drivers, Third Edition*. O'Reilly Media, Inc., février 2005.
- [4] Shourya P. Bhattacharya, Varsha Apte. *A Measurement Study of the Linux TCP/IP Stack Performance and Scalability on SMP systems*. Disponible sur <http://www.cse.iitb.ac.in/~varsha/allpapers/mypapers/comswarepaper.pdf> (consulté le 2 septembre 2008), 2005.