Projet RICM RobAIR

Années 2014-2015

Vivien MICHEL
Etienne VIALLET
Tararaina KLIPFFEL
Jeremy HAMMERER

Sommaire

Remerciements 2

I. Introduction 3

II. Objectifs 4

III. Organisation 5

A. L'équipe 5

B. Le projet 5

IV. Réalisation 6

A. RIOT-OS 6

B. Firefox OS 7

V. Conclusion 8

Bibliographie : 9

Références : 9

Remerciements

Nous tenons à remercier dans un premier temps, Didier Donsez, Amr Alyafi et Olivier Richard, pour nous avoir fourni le matériel nécessaire pour travailler dans les meilleures conditions possibles, et d'avoir été présents tout au long de la réalisation de notre projet. Nous les remercions aussi pour leurs aides et leurs conseils qui nous ont permis de continuer d'avancer malgré les difficultés rencontrées. Nous témoignons toute notre reconnaissance pour les expériences enrichissantes et intéressantes que nous avons connues durant ces 12 semaines de projet. Enfin, nous remercions les membres de la liste de développement RIOT qui ont pris de leur temps pour répondre à nos nombreuses questions.

I. Introduction

Ce travail s'inscrit dans le cadre de notre parcours à Polytech dans la filière *Réseaux Informatiques et Communication Multimédia*. Parmi les thèmes qui nous ont été proposés, c'est le projet RobAIR qui a suscité chez nous le plus d'intérêt. RobAIR est une plateforme robotique de téléprésence. Cela signifie qu'il est capable de se substituer à une présence, en intégrant des fonctions de téléguidages ainsi que de visiophonie. Nous pouvons lui imaginer divers domaines d'application. RobAIR pourrait permettre à un enfant hospitalisé de continuer sa scolarité avec ses camarades. Une personne à mobilité réduite pourrait aussi, par le biais de RobAIR, effectuer la visite guidée d'un musée. Ou encore, un chercheur pourrait depuis son bureau, participer à des conférences à l'autre bout du monde. RobAIR se différencie des plateformes existantes par l'utilisation de matériels et codes source ouverts, la rendant extensible et réduisant son coût par la même occasion.

Concernant nos motivations, elles venaient principalement de l'envie de travailler sur un projet concret et innovant, qui pourrait trouver une réelle importance dans la vie quotidienne. Nous pensions aussi que le domaine de la robotique serait différent des sujets informatiques sur lesquelles nous avons l'habitude de travailler. Enfin, nous trouvions intéressant de participer à un projet pluridisciplinaire, pour lequel différentes équipes d'étudiants se sont déjà relayés dans l'avancement.

Nous avons traité les différents aspects du projet selon une méthodologie respectant les principes du génie logiciel. Ce rapport explicitera cette organisation, ainsi que les différentes réalisations des objectifs qui nous ont été donnés.

II. Objectifs

Les objectifs de notre projet vont dans la continuité des contraintes générales du robot, qui sont sa modularité et son prix abordable. Ils peuvent être découpés de la manière suivante :

- Portage sur une carte STM32 Nucleo : développée par STMicroelectronics, cette carte a la particularité de pouvoir s'étendre facilement par l'ajout de nouveaux circuits « empilables ». Parmi ces extensions, on peut notamment citer la carte MEMS, intégrant un accéléromètre, un gyroscope, un magnétomètre ainsi que des capteurs d'humidité et de pression. Ces nouvelles fonctionnalités pourraient donc s'ajouter aux actuels capteurs ultrasons et lidar pour une meilleure maîtrise du dispositif. Un autre atout serait la carte bluetooth low consommation, qui assurerait une communication sans fil entre la STM32 et la tablette, et ce à des coûts modérés en énergie.
- Utilisation de RIOT-OS : un système d'exploitation open-source disposant d'un noyau restreint, et donc parfaitement adapté à une utilisation sur microcontrôleur STM32. Supportant les langages C et C++, cet OS est compatible avec de la programmation multitâche et en temps réel. Cependant, sa principale caractéristique réside dans son orientation vers l'internet des objets, ce qui s'applique spécialement à un robot connecté tel que RobAIR.
- Développement d'une application WebRTC pour Firefox OS : le système d'exploitation libre proposé par Mozilla Corporation. Conçu pour les smartphones et tablettes, il fait fonctionner des applications en format HTML, et supporte le CSS et le JavaScript. WebRTC étant une API de JavaScript dédiée à la communication en temps réel, Firefox OS permet d'élaborer facilement et efficacement une application destinée à la visiophonie.

III. Organisation

A. L'équipe

Notre équipe est composée de trois élèves ingénieurs d'option *communication multimédia* et d'un élève d'option *réseaux informatiques*. Nous collaborons également avec deux élèves de la filière *Informatique Industrielle et Instrumentation* dont le travail porte sur l'alimentation électrique de RobAIR. Enfin, nous avons été amenés à communiquer avec une équipe de l'ENSIMAG qui était en charge de ce projet l'année dernière.

En ce qui concerne notre groupe, nous pouvons affirmer que nous avons su rester soudés, ce qui nous a apporté l'inertie propice à un travail globalement plus efficace, et personnellement plus agréable. Étant originaires de parcours préparatoires différents, nous avons également chacun pu contribuer au projet de manière complémentaire selon nos forces et nos faiblesses.

B. Le projet

La prise en main et le développement de RobAir s'est déroulé sur 12 semaines. Les 6 premières semaines ont été consacrées à l'assimilation et la bonne compréhension du sujet. Ce travail préliminaire a principalement consisté en la lecture des documentations des cartes et capteurs ainsi qu'en l'étude des codes existants. En parallèle, nous avons suivi des tutoriels d'apprentissage interactif de C++, Python, HTML et CSS, pour nous familiariser avec les notions utilisées dans ce projet.

Les 6 autres semaines ont servi à la réalisation à proprement parlé. Le plus gros enjeu était d'assurer le bon fonctionnement de notre carte. Pendant ce temps, nous avons porté et amélioré une application utilisant WebRTC sur portable Firefox OS pour assurer le dialogue avec RobAir. Enfin, nous avons commencé les esquisses de portage du code Python pour Arduino à un code en C++ pour la carte STM32 Nucleo.

Ce sujet a été traité en étroite relation avec les cours de Génie Logiciel, ce qui nous a permis de structurer et consolider l'organisation de celui ci. Notamment avec l'utilisation d'une méthode agile "SCRUM". Cette méthode est composée de sprints concis et rapide, permettant une réactivité et une modularité du projet face aux demandes potentiels des clients. En effet, nous avons pu voir cet avantage lors des changements de carte au début du projet. De plus l'utilisation de gestionnaire de version "GIT" nous a permis de faciliter l'échange et la modification de nos fichiers sources.

IV. Réalisation

A. RIOT-OS

Nous avons tout d'abord testé la programmation sur la carte STM32 via mbeb[1], un environnement de développement en ligne, qui permet de créer des programmes C/C++ pour de nombreux types de carte telles que les STM32. Nous avons par ce biais, réussi à faire fonctionner les moteurs. Fort de cette première approche, nous nous sommes plongés dans le vif du sujet : RIOT.

Cependant, pour l'installation de RIOT-OS sur la carte STM32 Nucleo-L1, nous avons été confrontés à divers problèmes. Bien que nous arrivions à lancer des programmes de tests simples qui, par exemple, affichaient des messages dans la console, certains fichiers de configuration demeuraient incomplets pour la carte utilisée, nous empêchant d'exécuter des programmes plus élaborés. Nous avons eu comme première solution d'opter pour l'utilisation de mbed en tant que librairie externe sur RIOT. En effet, à travers les différents Makefile de l'OS nous avons essayé d'intégrer le code existant sur la plateforme de développement lors de la compilation du programme. Toutefois, les différences de conception et de mise en forme des structures de données (GPIO,PWM...) rendent incompatible l'utilisation simultanée de RIOT et MBED. De plus lors de l'édition de lien, certains fichiers n'étaient pas reconnus, faisant ainsi échouer la compilation.

Nous avons alors décidé de finir le fichier de configuration pour n'utiliser que des fonctions fournies par RIOT. C'est au cours de cette tâche que nous avons fait face aux plus de complications. Malgré l'aide de la communauté de RIOT et la lecture approfondie de la documentation fournie par ST[2][3], nous avons longuement été incapable de configurer RIOT correctement. En effet, le lien entre RIOT et les ports de la carte STM32 n'étant pas encore définis, nous ne pouvions lire de données d'entrées, ni fournir des informations en sortie. Plus précisément, nous nous sommes heurtés à des difficultés provenant de la mise en relation de la documentation très générale avec la syntaxe utilisée dans RIOT tout en ne sélectionnant parmi toutes ces informations que celle qui étaient nécessaire à la configuration voulue.

Un autre problème est survenu lorsque, voulant examiner avec plus de précision ce qui se passait lors de l'exécution du code, nous avons décidé d'utiliser un débogueur nommé OpenOCD. Lors du transfert d'un simple programme "Hello World" avec cet outil la carte STM32 a été irrémédiablement endommagée. En effet, nous ne pouvions plus transférer de programmes, et le processeur de la carte surchauffait. Nous pensons que celui-ci demeurait bloqué dans une boucle infinie.

Après de nombreuses tentatives infructueuses, nous avons finalement réussi à compléter le fichier de configuration “periph_conf.h” de la carte STM32 Nucleo-l1. Ce fichier est crucial pour la communication entre la carte et ses périphériques. Il permet notamment au shield Ardumoto de recevoir des signaux en impulsion modulée afin de faire tourner ses moteurs à différentes vitesses. Ceci est pour nous une réelle réussite assurant la pérennité de notre projet. En effet, les personnes qui travailleront plus tard sur ce projet auront un fichier de configuration fonctionnel. Elles n’auront plus qu’à porter et améliorer le code déjà existant sur RIOT-OS.

B. Firefox OS

Cet objectif demandait de réutiliser la page web déjà existante pour en faire une application destinée à tourner sur Firefox OS. Comme ce système d’exploitation utilise des sources en HTML pour ses programmes, il nous restait à lui déclarer notre application par le biais d’un fichier “manifest”. Nous en avons également profité pour revoir l’intégralité du code CSS, afin de proposer une interface adaptée à une utilisation sur smartphone. Nous n’avons en revanche pas réussi à vérifier les fonctionnalités webRTC réalisées par d’autres élèves l’année dernière. Même si nous sommes parvenus à lancer le serveur JavaScript de l’application en suivant les indications précisées dans le README, il nous a été impossible d’interconnecter deux machines.

V. Conclusion

Notre bilan vis-à-vis de notre progression est pour le moins mitigé. Il nous a permis de mettre en œuvre certaines connaissances théoriques vues au cours de notre formation. C'était le premier projet pour lequel nous avons clairement défini les contraintes à respecter, afin de mettre en lumière les différentes tâches à accomplir et nous permettre d'établir une organisation temporelle de ces dernières. Nous avons aussi eu l'occasion de de nous adresser à une communauté de développeurs, ce que nous aurons certainement à refaire dans notre future carrière d'ingénieur. Enfin, ce projet a fait appel à des compétences techniques, certaines qui étaient déjà acquises, mais également d'autres que nous avons dû découvrir, nous forçant à adopter une attitude proactive. Cependant, nous avons aussi quelques regrets à exprimer. En effet, notre travail se résumé en grande partie à nos tentatives d'adapter RIOT sur la carte que nous possédions, pour laquelle l'implémentation était incomplète. En plus de nous avoir coûté beaucoup de temps, elle a aussi été un grand frein pour les autres objectifs, rendant caduc notre planification.

Bibliographie :

STMicroelectronics :
http://www.st.com/

RIOT :
http://www.riot-os.org/

Firefox OS :
https://www.mozilla.org/fr/firefox/os/

Références :

[1] mbed :
http://developer.mbed.org/

[2] DataSheet :
http://www.st.com/web/en/resource/technical/document/datasheet/CD00277537.pdf

[3] Reference Manual :
http://www.st.com/web/en/resource/technical/document/reference_manual/CD00240193.pdf