JOG : une approche haut niveau des systèmes embarqués via Armadeus et Java

Olivier Reynet[1], Jean-Christophe Le Lann[1], Benoît Clément[1]

[1]ENSIETA, 2, rue François Verny, 29200 Brest - France

olivier.reynet@no-spam@ensieta.fr

Resumé

JOG est un robot mobile indoor dédié à l'enseignement des systèmes embarqués à base de processeur ARM, du système d'exploitation Linux et d'une machine virtuelle Java. JOG permet d'aborder le développement d'un système embarqué complet avec un langage de haut niveau, Java, sans pour autant faire abstraction des aspects matériels du système. Une API Java Jarmadeus a été développée afin de rendre possible ce lien matériel-machine virtuelle. Elle a été testée lors d'un module d'enseignement présenté sous forme de compétition de robots aux étudiants.

Mots-clés: ARM, FPGA, Java, Linux, capteurs, automate, conception

1 Introduction

JOG est un robot mobile indoor dédié à l'enseignement des systèmes embarqués à base de processeur ARM, d'un FPGA, du système d'exploitation Linux et d'une machine virtuelle Java. Ces quatre éléments sont représentatifs des technologies phares des systèmes embarqués actuels et sont disponibles simultanément sur la plateforme open source Armadeus.

1.1 Vocations de JOG

JOG a deux vocations initiales. La première vocation de JOG est pédagogique : il s'agit de présenter les concepts des systèmes embarqués à des étudiants de première année d'école d'ingénieur. Ceux-ci ont des notions dans les domaines de l'automatique, de la programmation Java et de l'électromécanique. Cependant, ils ne connaissent pas le langage C et n'ont quasiment aucune notion d'électronique numérique.

La seconde vocation de JOG est liée à la recherche : JOG est une plateforme peu onéreuse et facile à construire. Plusieurs exemplaires de JOG permettront d'étudier des comportements multi-agents, dans les domaines de la modélisation et de la localisation.

1.2 Origine de JOG et approche

L'approche des systèmes embarqués adoptée ici est résumée sur la figure 1. À l'origine de cette approche et de la conception de ce robot, plusieurs constats d'enseignants-chercheurs :

1. Pour la majorité des étudiants, il est important de concrétiser les apprentissages au travers de projets motivants, transverses et cohérents avec leurs enseignements précédents. La robotique fournit un terrain idéal tant au niveau de la concrétisation qu'au niveau de la transversalité. C'est pourquoi nous avons présenté le module aux étudiants comme un *challenge* de robotique.

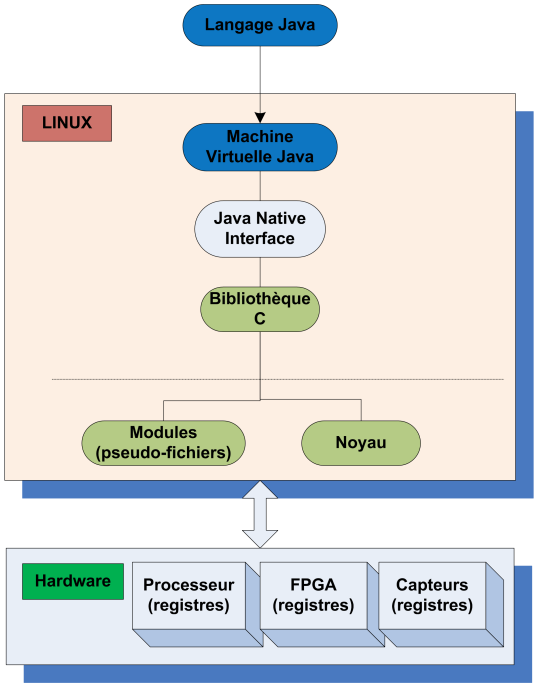

Figure 1: Approche JOG des systèmes embarqués : les étudiants ne programment qu'en Java mais accèdent néanmoins aux registres des périphériques à la manière des langages de bas niveau.

2. Les étudiants doivent appréhender les concepts les plus importants, sans pour autant se noyer dans l'aspect technique. Il est donc essentiel de leur masquer les difficultés techniques, sans pour autant leur ôter le contrôle et la vision matérielle du système. Concrètement, il nous apparaît important que les étudiants consultent les datasheets des composants électroniques qu'ils ont à programmer, et ne s'appuient pas sur une quelconque interface graphique qui masque totalement le composant. Aussi avons-nous décidé de créer et de fournir aux étudiants une interface logicielle souple constituée d'une API Java utilisant des bibliothèques C et Java Native Interface (JNI). Cette API *mandataire* préserve la vision bas niveau *registre* des matériels électroniques tout en simplifiant la lecture et l'écrire sur ces registres.

3. L'importance croissante de la conception conjointe dans les systèmes embarqués rend nécessaire une approche des systèmes embarqués dans toute leur *verticalité*, et non pas uniquement par le bas ou par le haut niveau comme cela est souvent présenté[1]. Cela signifie que la *modélisation des comportements* tout comme la *programmation des éléments matériels* de bas niveau doivent trouver leur place dans l'enseignement des systèmes embarqués.

La section suivante présente en détail les éléments constitutifs matériels et logiciels de JOG. La troisième section décrit l'API Java Jarmadeus dont les étudiants de première année se sont servis pour réaliser le JOG Challenge. La quatrième partie décrit les différents contextes pédagogiques qui mettent en œuvre le JOG.

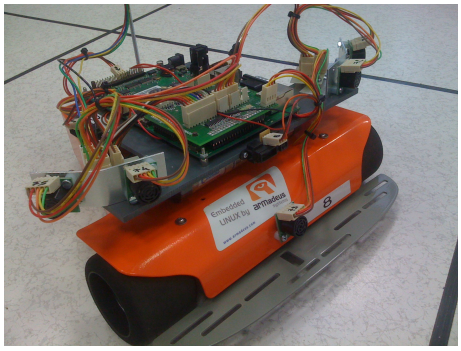

Figure 2: Photographie du robot JOG

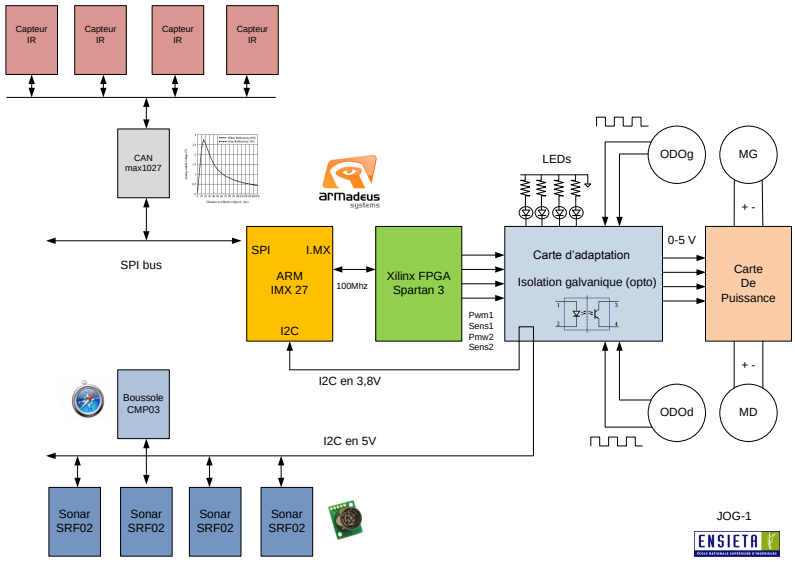

Figure 3: Synoptique du robot JOG

2 Anatomie d'un JOG

2.1 Mécanique

Le robot JOG (cf. figure 2) est très simple sur le plan mécanique. Il s'agit principalement d'une plateforme Stinger, disponible sur internet au prix de 100 euros environ. Il comporte un châssis en aluminium, 2 moteurs DC, 2 odomètres, 1 roue folle et 1 interupteur. Les batteries 12V permettent d'obtenir une autonomie d'environ 1 heure. La valeur totale d'un JOG est d'environ 800 euros, ce qui reste très raisonable par rapport aux capacités de ce robot.

2.2 Carte électronique, capteurs et bus

L'importance croissante de Linux et des logiciels libres pour l'embarqué rend la carte **open source Armadeus**[2] relativement incontournable pour des projets innovants dans le domaine. Des concur-

[1] De nombreuses sociétés proposent des solutions clés en mains, avec interfaces graphiques exceptionnelles : Lego MINDSTORMS NXT, Robotino de Festo, POB Robotics Suite. . .

[2] http://www.armadeus.com/francais/ et http://www.armadeus.com/wiki/

rents existent, notamment la carte Fox[3], ou la Beagle Board[4]. Les avantages de l'Armadeus, en plus d'être un projet open source, sont notamment d'utiliser Buildroot, Busybox, Binutils, Linux et GCC.

Grâce à la carte Armadeus APF27 et la carte de développement APF27-devfull, le JOG dispose de nombreuses interfaces, dont deux bus I2C, deux bus SPI, et des liens série. Des capteurs faibles coûts ont été connectés sur le bus I2C (cf. figure 3) : 5 télémètres sonar I2C SRF02 (20-200 cm) et une boussole I2C CMP03. Par ailleurs, la carte de développement comporte un convertisseur analogique numérique Max1027. Quatre détecteurs d'obstacles IR analogiques GP2D120 (0-40cm) ont donc été connectés sur les voies 2,4,5 et 6 du Max1027. Une carte d'adaptation a été developpée, afin de réhausser les tensions des bus pour les capteurs.

Ces capteurs sont tous faibles coût et leurs performances sont donc très moyennes. Pour plus de précision dans l'exploration de l'environnement, un laser Hokuyo peut facilement être interfacé avec la carte Armadeus. Il est également possible de connecter une centrale inertielle, par exemple via le port USB de l'APF27 ou un GPS.

2.3 Linux, modules Armadeus et bibliothèque as_devices

Le système d'exploitation de la carte Armadeus est Linux (noyau 2.6.29.6). Le système de fichier est au format Ubifs et le Bios est u-boot. Le temps de démarrage est d'environ 10 secondes. Busybox permet ensuite de commander simplement le système d'exploitation. Des modules spécifiques à Armadeus peuvent être chargés pour faciliter l'accès aux périphériques, notamment pour piloter le convertisseur analogique numérique ou pour gérer les GPIO. À noter qu'il existe une bibliothèque as_devices au sein du projet Armadeus, bibliothèque qui a inspiré notre approche : il s'agit d'une bibliothèque en C et C++ qui permet d'interfacer aisément tous les périphériques de la carte de développement.

2.4 FPGA pour des périphériques à la demande

Le FPGA est utilisé comme un fournisseur de périphérique à la demande[5]. En l'occurence, nous avons choisi de créer 4 PWM (2 pour les moteurs, 2 pour des servomoteurs pour des capteurs/actuateurs auxiliaires), un timer hardware, un registre qui pilote 4 diodes et un système de comptage d'impulsions provenant des odomètres des moteurs du JOG. Pour concevoir ce système, nous avons utilisé GHDL, GTKWave, make et les outils de synthèse Xilinx.

L'interaction avec le FPGA a été très appréciée par les étudiants : ceux-ci ont réalisé l'intérêt d'avoir un système capable de fonctionnalités de haut niveau et indépendant du processeur. Le système de comptage des impulsions de l'odomètre a permis aux étudiants de comprendre le fonctionnement interne du FPGA. Le registre pilotant des diodes a renforcé l'acquisition des concepts de l'électronique numérique de base (base 2 et 16, masques, opérations logiques) et l'usage des opérateurs bits à bits.

3 Jarmadeus, une API Java qui facilite l'accès au matériel

3.1 Talker, un intermédiaire à qui parler

D'une manière générale, nous avons choisi d'utiliser le concept de *mandataire* pour créer une interface Java-matériel. Ce mandataire a pour mission de réaliser le dialogue avec des pseudo-fichiers Linux créés par des modules Linux (par exemple pour le convertisseur analogique numérique),

[3] http://foxlx.acmesystems.it/
[4] http://beagleboard.org/
[5] Cette vision est également portée par le projet POD sur le wiki Armadeus.

avec un bus I2C ou avec le FPGA. Un package Java formé de classes de type Talker (cf. figure 4) a été développé, dans lequel chaque classe propose une interface simple d'écriture et de lecture sur des systèmes particuliers. Les classes Talker ne suffisent pas pour programmer un composant, car il est nécessaire de transmettre les bonnes commandes, mais elles facilitent grandement la communication avec le composant.

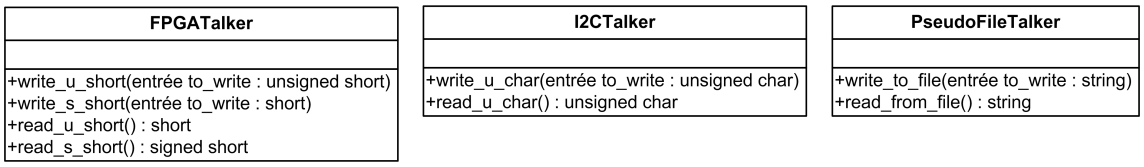

Figure 4: Package Jarmadeus : intefaces des classes Talker

3.2 Types de données, bus I2C et JNI

Chaque classe Java n'est en fait qu'un masque qui cache l'appel à des fonctions écrites en langage C et utilisant JNI. Ces interfaces nécessitent un transtypage des données, les types de données en C et en Java ne se recouvrant d'aucune manière. Cela pose d'ailleurs des problèmes épineux, Java ne possédant pas, par exemple, de type utilisable pour les données 8 bits non signés. Or, sur un bus I2C, c'est le type de donnée qui circule le plus souvent. Pour simplifier, nous avons fait utiliser aux étudiants des char Java (16 bits non signés) et des short (16 bits non signés). Ces types sont modifiés puis rétablis par les fonctions écrites en C. Les registres du FPGA étant codés sur 16 bits, ces problèmes ne sont pas apparus pour ce périphérique.

3.3 Dialogue ARM-FPGA : memory mapping

Le dialogue avec entre le processeur et le FPGA s'effectue via un memory mapping en langage C. La zone mémoire représentant les registres du FPGA est synchronisée avec le contenu réél des registres. L'accès au FPGA est donc très rapide, même en Java grâce à JNI.

3.4 Pseudofichiers Linux et Java

Les pseudo-fichiers Linux sont accessibles comme n'importe quel fichier depuis Java. Cette classe aurait donc pu être écrite par les étudiants. Elle renforce le concept *tout fichier* Linux. D'une manière plus générale, de nombreux fichiers ont été ouverts et fermés périodiquement au travers de ces mandataires. L'appel aux fonctions C étant momentané, on ne peut pas sauvegarder de descripteur de fichier d'une manière simple tout au long de l'utilisation des fichiers, sans que ce soit apparent pour les étudiants et sans modifier profondément les classes Talker. Néanmoins, l'accès Java au système de fichier étant très rapide, JOG ne souffre pas de cette limitation.

4 JOG à l'ENSIETA

4.1 JOG Challenge

Ce JOG challenge a été fédérateur pour les enseignants et les étudiants. Inspiré des pratiques de la pédagogie active (learning by doing), il a été réalisé en 7 séances de 4 heures par des groupes de 6 étudiants. Il a inspiré de nombreuses réflexions aux étudiants au niveau de la démarche scientifique, des capacités de l'électronique numérique, de la perception de l'environnement avec

des capteurs, de la conception d'un automate et du travail en équipe. La compétition a été le moteur du développement des étudiants. Le challenge s'est déroulé en deux phases :

1. la qualification : les étudiants devaient prouver qu'ils étaient capables de commander chaque fonctionnalité de JOG de manière indépendante. Cette épreuve était un préalable nécessaire au challenge.

2. le challenge en lui-même.

4.1.1 Figures imposées

Chaque groupe d'étudiants devaient impérativement présenter à l'issu du projet :

1. un descriptif détaillé de la mission que devait réaliser le robot,

2. la machine à état du robot pour la mission en question,

3. l'architecture des classes utilisées (avec les patrons de conception éventuels, State et Strategy ont été très utilisés),

4. au moins une régulation simple du JOG (à partir des odomètres, de la boussole. . .),

5. les fichiers journaux de la mission du robot.

Par ailleurs, chaque étudiant a dû produire un compte rendu de deux pages maximum, synthèse du projet selon le rôle qui'ils avaient choisi : chef de projet, expert comportement, expert perception et propulsion, expert test, expert méthode ou expert intégrateur.

4.1.2 Figures libres et expériences

Chaque équipe a choisi la nature de la mission que devait réaliser le robot : sortie de labyrinthe, créneau, suivi de mur, suivi de trajectoire. . . Certains ont également présenté des fichiers journaux évolués avec interpréteur. D'autres ont perfectionné les séquences de test. Les challenges ont été réalisés à l'intérieur de l'ENSIETA ou sur un terrain de basket extérieur. Lors de la réalisation du challenge, les étudiants se sont confrontés à la réalité des données des capteurs. Les perturbations de la boussole à l'intérieur de l'ENSIETA, les imperfections de mesure des télémètres ou le bruit des capteurs optiques ont été très déstabilisant pour des étudiants souvent pétris de certitudes quant aux mesures physiques et d'une naiveté parfois troublante. La démarche scientifique abstraite et désincarnée des élèves de classe préparatoire a dû faire place à une démarche plus concrète, ou le bon sens et l'observation sont les meilleurs guides. D'une manière générale, les étudiants ont compris que le JOG Challenge était pour eux l'occasion de concrétiser leur efforts en Java en réalisant, souvent pour la première fois de leur vie, le contrôle d'un système complet.

4.2 JOG en temps réél, en modélisation

En plus de l'initiation aux systèmes embarqués de première année, JOG est utilisé de le cadre de projets des étudiants en troisième année. Par ailleurs, l'ENSIETA forme également des ingénieurs par alternance dans le domaine des systèmes embarqués. Dans ce cadre, trois utilisations de JOG vont voir le jour d'ici à la fin 2010. La première s'inscrit dans le cadre de l'enseignement du temps réél pour l'embarqué en langage C et avec Xenomai. La seconde cherche a créer un système de codesign effectif basé sur la carte Armadeus. La dernière se sert de JOG pour concrétiser les concepts de l'automatique.

5 Conclusion

JOG est un outil pédagogique simple de conception, peu onéreux et multifonction. Il a le mérite de présenter aux étudiants un système complet, sans masquer la réalité technologique des composants électroniques et en proposant une API simple d'utilisation fondée sur le principe du mandataire. Il peut être utilisé en traitement du signal, en automatique, en programmation temps réél ou en électronique numérique.