MINAmI
Programmation par objets d'applications d'ambiances intelligentes avec LabVIEW OOP

Yann FALEVOZ

Assystem
10, Chemin du pré carré
38240 Meylan
yann.falevoz@gmail.com

Thierry PORCHERON

Hager
246, rue du pré de l'Horme
38920 Crolles
t.porcheron@hager.fr

Résumé : Cet article présente les différentes étapes du développement d'un démonstrateur de système domotique réalisé dans le cadre du projet européen MINAmI, qui vise à la mise en œuvre d'applications d'ambiances intelligentes, centrées autour du téléphone portable. Ce projet a été l'occasion de prendre en main la programmation objet sous LabVIEW et de découvrir ses points forts et ses faiblesses.

Mots clés : Programmation objet, LabVIEW OOP (LVOOP), System on cheap (SOC), très faible consommation, systèmes distribués, SOC, protocole RF

INTRODUCTION

Le projet MINAmI aborde les défis liés à la mise en œuvre d'applications d'ambiances intelligentes (AMI), où le téléphone mobile agit comme une interface. Les travaux novateurs de MINAmI sont axés sur des démonstrateurs concrets d'AmI (WP7 Figure 1). Les principaux partenaires de ce projet sont Nokia, ST, le CEA LETI, le CSEM, VTT...

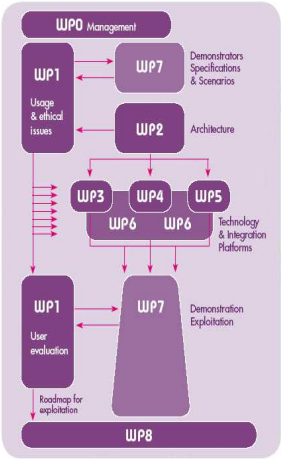

Figure 1 : Découpage du projet en "work package".

L'un de ces démonstrateurs, développé par le service recherche de la société Hager, vise à offrir des services de types domotique, sécurité et « home care », dans une optique délibérément non invasive, en se reposant sur les informations données par un ensemble de capteurs répartis en réseau dans un logement. Ces capteurs (détecteurs infrarouges, détecteurs d'ouverture, capteurs de vision intelligents, ou encore accéléromètres), regroupés au sein de clusters, communiquent entre eux et avec le nœud central (PC) par lien RF (protocole très faible consommation). Un laptop, connecté au nœud central, permet à l'installateur de configurer et de superviser le système.

Le développement du Nœud Central et du Laptop s'articulant autour de LabVIEW et nécessitant une expérience non négligeable dans ce domaine, il a été décidé de faire appel à la Société Assystem pour assurer ce développement.

POURQUOI LABVIEW ?

En ce qui concerne le choix des langages utilisés au niveau du nœud central et du Laptop, plusieurs impératifs techniques entraient en jeu :

- Gestion aisée des couches basses (Modem RF , ethernet, …).
- Rapidité et efficacité de l'écriture des logiciels.
- Démonstrateur aisément maintenable et évolutif.
- Applicatif utilisateur (scenarii), côté nœud central, écrit dans une sémantique haut niveau (grafcet, XML, …), adaptée à l'utilisateur final, et facilement supportée par le « langage pivot » retenu.
- Souhait d'implémenter, à terme, le nœud central sur une cible compacte, de type Gumstix.

LabVIEW, langage graphique générique et supportant l'objet, répondait a priori à ces attentes. Il a donc été décidé de l'utiliser, aussi bien au niveau du Nœud Central que du Laptop.

L'un des objectifs prioritaires de l'équipe Recherche de la société Hager étant, par ailleurs, de tendre vers le développement rapide de maquettes fonctionnelles, en amont de la phase Marketing, ce projet lui permettait également de tester LabVIEW en tant qu'outil de développement de telles maquettes.

DES TECHNOLOGIES INNOVANTES

Parmi les technologies mises en œuvre au sein du démonstrateur « Ambiant Sensors for Friendly Home Applications », trois sont particulièrement innovantes.

Le Vision Sensor

Véritable rétine artificielle développée par le CSEM (Suisse), associée à un DSP (Blackfin Figure 2), elle se démarque des imageurs traditionnels par sa très grande dynamique (7 décades), l'extraction directe de contrastes orientés (Figure 3) au niveau de la rétine (pas de 5° – 20 images/s), ainsi que sa faible consommation (300mW). Sa puissance de traitement permet de garantir un mode non invasif (transmission des informations essentielles uniquement – selon les cas : position du mobile, identification du mobile, chute, … - sans transmission d'images).

Figure 2 : De gauche à droite : la carte DSP, la carte modem et la rétine.

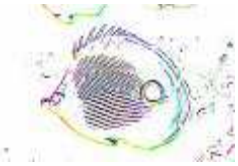

Figure 3 : Exemple d'image en contraste orienté.

Le nano-accéléromètre

Cet accéléromètre trois axes (Figure 4) développé par le LETI (Grenoble) présente des caractéristiques exceptionnelles en terme de consommation (100µA par axe), tout en offrant une sensibilité intéressante (5mG) pour les domaines d'application visés (anti-intrusion, confort, maintenance à domicile). La petite taille de la puce autorise, de plus, de faibles coûts (< 1 €).

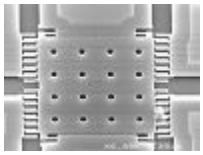

Figure 4 : Le nano-accéléromètre.

L’interface RF

Figure 5 : Le modem.

Il s’agit d’un SOC (System On Chip, Figure 5), développé par Hager dans une optique systèmes distribués (Mesh Flow). Il se caractérise d’un point de vue RF par des caractéristiques très intéressantes en termes de consommation (3mA sous 1V) et d’intégration (bandes ISM 400MHz et 800MHz). Le protocole très faible consommation supporté par ce SOC (propriété d’Hager) est compatible avec les contraintes d’autonomie (> 5 ans) et de temps de réponse (100ms) propres aux systèmes Hager Security.

CAHIER DES CHARGES

Ce démonstrateur doit se présenter comme le prototype d'un produit potentiel. Il doit donc présenter toutes les fonctionnalités nécessaires à son installation et à son utilisation. L'objectif est de faire en sorte que ces fonctionnalités soient les plus conviviales et les plus simples d'utilisation possible. En ce qui concerne LabVIEW, le démonstrateur comprend trois programmes distincts.

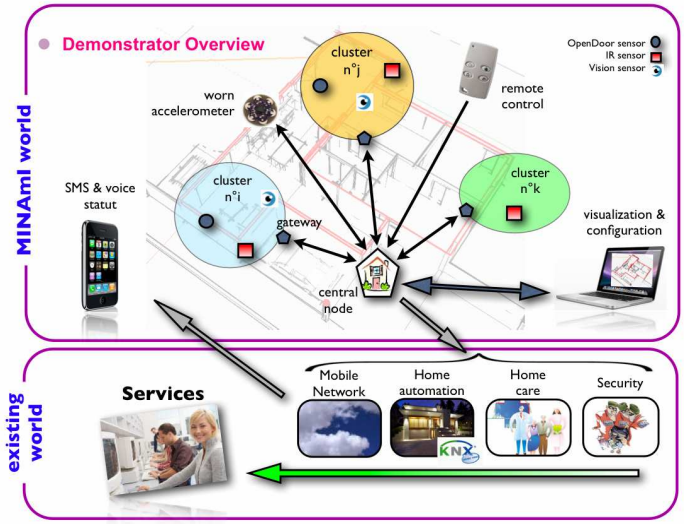

Figure 6 : Relation entre les différents éléments du démonstrateur.

Le laptop

Ce programme concerne l'installateur. Il lui permet de configurer et de superviser le système en temps réel. La configuration se fait par la définition grossière du logement (sans respecter les échelles), l'implantation des capteurs et leurs calibrations. Cette dernière étape permet notamment de faire le lien entre le plan et la pièce réelle. Pour ce qui est de la supervision, le programme laptop retranscrit sur le plan, en temps réel, l'état des capteurs tels qu'ils lui sont transmis par le nœud central.

Le nœud central

C'est le chef d'orchestre du système. Il reçoit les informations émises par les capteurs (spontanément ou sur demande) et les enregistre. Il peut aussi envoyer des ordres aux capteurs, ainsi qu’aux effecteurs. Il doit être programmable, c'est-à-dire que l'installateur doit pouvoir indiquer au système les actions qu'il doit effectuer lorsque certaines conditions sont remplies. Par exemple, allumer la lumière des escaliers si la porte permettant d'y accéder est ouverte et qu'une personne s'en approche, mais pas si c'est un chat; ou appeler les secours si une personne est tombée et qu'elle ne répond pas aux sollicitations du système. Si le laptop lui est connecté, il doit lui remonter les informations émises par les capteurs pour la supervision.

L'espion

Ce programme ne fait pas, à proprement parler, partie du démonstrateur. Il est cependant nécessaire à la mise au point du modem et du nœud central. Il doit permettre de visualiser et d'enregistrer tous les messages échangés entre les capteurs et le nœud central. Il doit, de plus, permettre d'envoyer des messages en se faisant passer pour le nœud central ou pour n'importe quel capteur.

Points de difficulté

De par la nécessité de devoir dessiner des plans, le programme « laptop » fait une utilisation relativement atypique de LabVIEW et il a fallu étudier la meilleure façon de répondre à ce besoin.

L'aspect programmable du nœud central est l'un des plus gros points de difficulté. En effet, il implique non seulement d'inventer un langage permettant de décrire les conditions et les actions, mais aussi de programmer un interpréteur de ce langage dans le nœud central.

Certaines optimisations au niveau du système d'exploitation sont à envisager pour rendre le nœud central compatible avec les temps de réponse, très courts, requis par le modem, LabVIEW-RT n'étant pas envisagé pour des raisons de budget.

Ce projet a démarré peu de temps après l'apparition de la programmation objet dans LabVIEW. Il a été l'occasion de découvrir LVOOP, ses qualités et ses points faibles.

CONCEPTION

La première étape de conception a été de réaliser un diagramme de classe fixant l'architecture du logiciel concernant la gestion du plan et des capteurs. Pour rendre la programmation la plus naturelle possible, ce diagramme (~40 classes) reflète autant que possible le monde réel. Il indique ainsi que la classe « plan » se compose d'étages, que ces étages se composent eux mêmes de pièces, que les escaliers sont un type de pièce particulier, etc... Cette structure (Figure 7) a permis de tirer pleinement profit de l'héritage et du polymorphisme introduit avec la programmation objet. Par exemple, les ouvertures d'une pièce peuvent être gérées comme un seul tableau d'ouvertures, qu'il s'agisse de portes ou de fenêtres.

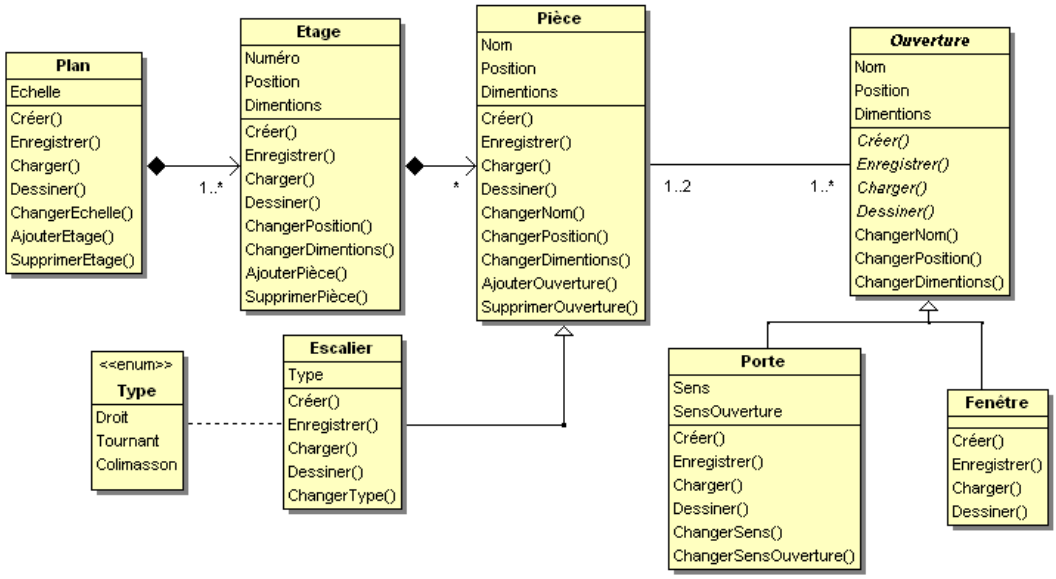

Figure 7 : Extrait du diagramme de classe.

Une seconde étape a été de définir le protocole de dialogue entre les capteurs et le nœud central. Ce protocole a bien sûr dû tenir compte des informations susceptibles d'être émises

par chaque capteur, le plus complexe d'entre eux étant le capteur de vision intelligent, qui donne une grande quantité d'informations.

Concernant le langage permettant de programmer le comportement du nœud central, le choix s'est porté sur une base de langage XML (Figure 8). Ce dernier étant un métalangage de description de données, il est tout à fait adapté pour décrire des conditions et les actions qui leurs sont associées.

```
<Event>
    <Condition>
        <BooleanOp Operator="Yes">
            <Flag SensorName="Remote control 0" FlagName="BP1" ValueCriteria="0"/>
        </BooleanOp>
    </Condition>
    <Action Name="WakeUp IR" Mode="Alarme" Type="Command" Param="IR 0;1;2100000000"/>
    <Action Name="WakeUp D0" Mode="Alarme" Type="Command" Param="Opening Sensor 0;1;2100000000"/>
    <Action Name="WakeUp IR" Mode="Confort" Type="Command" Param="IR 0;1;2100000000"/>
    <Action Name="WakeUp D0" Mode="Confort" Type="Command" Param="Opening Sensor 0;1;2100000000"/>
    <Action Name="WakeUp IR" Mode="MAD" Type="Command" Param="IR 0;1;2100000000"/>
    <Action Name="WakeUp D0" Mode="MAD" Type="Command" Param="Opening Sensor 0;1;2100000000"/>
</Event>
```

Figure 8 : Exemple d'évènement : élément unitaire du fichier programme se composant d'une condition et d'un ensemble d'actions.

Une dernière étape a été de trouver des pistes pour l'optimisation du système d'exploitation pour répondre à la contrainte de temps de réponse du modem. Le programme nLite a été utilisé pour alléger Windows au maximum et l'appel à la librairie « kernel32.dll » dans le programme a permis de passer LabVIEW en priorité « temps réel » (Figure 9).

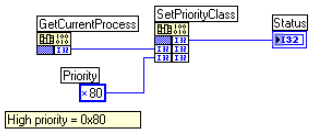

Figure 9 : Appel à la fonction SetPriorityClass de la librairie "kernel32.dll".

DEVELOPPEMENT

La priorité étant mise sur l'espion pour la mise au point du modem, le premier point a été de réaliser un driver pour ce dernier. Puisque le projet a initialement débuté sous LabVIEW 7.1, ce driver a été développé à l'aide du GOOP Wizzard 1 d'Endevo. Il a permis d'avoir très rapidement un programme espion opérationnel (Figure 10). Celui-ci comprend une boucle de gestion des évènements, une boucle de gestion du modem et une boucle de gestion des logs.

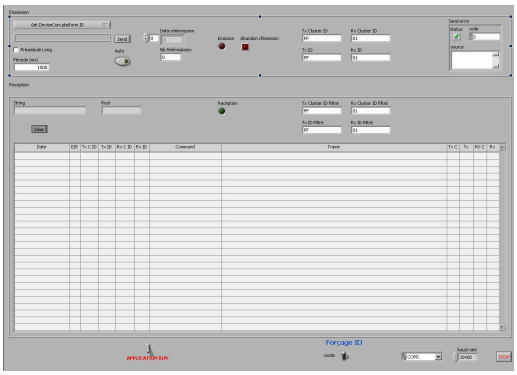

Figure 10 : Interface graphique de l'espion.

Une grande partie du temps de développement a été consacrée au développement du programme laptop et notamment à son éditeur de plan (Figure 11). Ce programme est construit suivant l'architecture « queued state machine » avec une boucle d'interception des événements de l'interface qui transmet les actions à effectuer à une seconde boucle. Deux boucles supplémentaires permettent le dialogue avec le nœud central, l'une pour l'émission, l'autre pour la réception. La sauvegarde des plans se fait dans des fichiers XML grâce à la librairie MSXML15. Les aspects héritage et polymorphisme qu'apporte la programmation

objet ont été largement utilisés. Une vidéo de présentation de l'interface de ce programme peut être visionnée à l'adresse : http://www.youtube.com/watch?v=EMQZk86BmXE.

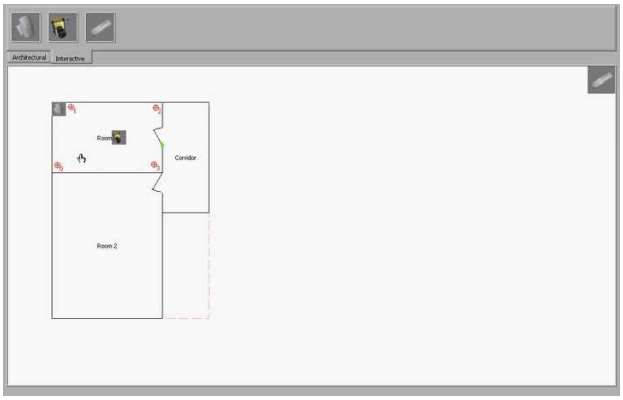

Figure 11 : Interface graphique du programme "Laptop".

Le programme du nœud central réutilise toutes les classes développées pour la gestion du plan et des capteurs. Il ne possède pas d'interface car il est censé fonctionner de manière autonome. C'est la face avant du laptop, lorsqu'il lui est connecté, qui lui sert d'interface. Il se compose d'une boucle de gestion du modem, une boucle d'interprétation des trames reçues, une boucle de gestion des logs, une boucle de communication avec le laptop et une boucle d'exécution du programme applicatif.

BILAN ET PERSPECTIVES

L'utilisation de LVOOP dans ce projet a été un réel plus. Cela a permis d'avoir une architecture très intuitive et donc de faciliter la programmation. Quelques points se sont toutefois avérés pénalisants.

La plupart des dialogues entre le laptop et le nœud central font appel au VI « Flatten to XML ». Le fait que ce VI ne soit pas compatible avec les classes a posé des problèmes pour le transfert du plan. Il a ainsi fallu faire une sauvegarde préalable dans un fichier temporaire, puis transmettre le contenu de ce fichier. Ce point a semble-t-il été corrigé dans la version 8.6 de LabVIEW.

Le fait de ne pas pouvoir manipuler les classes par référence s'est révélé particulièrement gênant sur certains objets. Prenons l'exemple d'une porte reliant deux pièces entre elles. Au moment de la conception, il paraît naturel de définir la classe « porte » comme classe associative, puisqu'elle fait le lien entre deux instances de la classe « pièce ». La porte est ainsi partagée par les deux pièces. Chacune d'entre elles sait où trouver la porte, mais elle n'est constitutive d'aucune d'entre elles. En d'autres termes, chacune des pièces a une référence à la porte comme attribut, mais pas la porte elle-même. Une telle conception permet de faire des modifications sur la porte depuis l'une ou l'autre des pièces sans se soucier de la seconde. LabVIEW n'autorisant pas à manipuler les classes par référence, il a fallu mettre une porte dans chaque pièce. Toute modification de la porte par l'une des pièces implique donc la modification de la porte associée dans l'autre pièce, ce qui complique beaucoup le programme. La « GOOP development suite » d'Endevo permet de gérer facilement l' « openGOOP » ou l'« Endevo GOOP 3 », qui sont des alternatives à la structure native des classes sous LabVIEW, permettant de manipuler, soit des objets par référence, soit des objets dont le seul attribut est une référence à un cluster d'attributs, et donc de palier au problème rencontré. La mauvaise connaissance de cet outil a conduit à son écartement du projet, de peur que les limitations de la « Community Edition » ne soient trop contraignantes.

L'une des perspectives est d'embarquer le programme du nœud central sur une cible beaucoup plus compacte (Gumstix par exemple) à l'aide du module LabVIEW Embedded.