Projet IMA4 2015-2015

ROBOCUP 2016

Interface de communication

PAYELLE VIANNEY

Encadrants : Vincent Coelen , Rochdi Merzouki

Remerciements

Je remercie mon encadrant principal : Mr. Vincent Coelen pour pour son soutien moral et technique tout au long de ce projet, ainsi que Mr. Rochdi Merzouki pour avoir soutenu le projet.

Je remercie aussi Sandra Hage Chehade, Valentin Vergez et Thomas Danel pour leur implication dans le projet : choix d'architecture, aide morale et technique.

Le projet étant réalisé dans le cadre de l'ARPL, je tient à remercier tout les autres membres de l'association : Othman Lakhal, Cyril Smagghe et Gwendoline Smagghe, pour leur implication au sein de celle-ci.

Je remercie aussi Polytech Lille et l'université de Lille 1, le département IMA , ainsi que le laboratoire CRIStAL, en tant que sponsors officiels de l'association.

Sommaire

Introduction

Dans le cadre de la RoboCup Logistics League, j'ai rejoins l'équipe PyroTeam pour l'Open German de 2015, équipe qui est aujourd'hui intégrée à l'Association de Robotique de Polytech'Lille : ARPL.

Cette compétition sponsorisée par la société FESTO a pour but de promouvoir le développement, la recherche, et l'utilisation de robots mobiles autonomes en milieu industriel.

Dans ce cadre, deux équipes de trois robots s'affrontent dans une compétition de production industrielle optimisée, en terrain partiellement connu, de manière autonome, en communiquant avec un gestionnaire et des machines de production.

Dans le cadre de notre équipe, le système fonctionnel des robots est géré par le Robot Operating System (ROS), qui réalise la gestion différents processus et la communication entre ceux-ci au sein du robot.

Cependant, dans le cadre de la compétition, une phase de préparatoire de seulement cinq minutes est accordée pour paramétrer et démarrer les différents processus des robots. Ce délai est très court et il n'existe pas d'interface avec ROS répondant à nos attentes pour cette tâche.

C'est pourquoi il a été décidé de développer celle-ci, de manière personnalisée pour la Robocup d'un point de vue ergonomique, tout en restant suffisamment générique pour pouvoir l'utiliser avec d'autre robots, dans le cadre d'une autre compétition, un autre projet ou un programme pédagogique.

Présentation du projet

1 La RoboCup

La RoboCup est un tournoi international de robotique.

Son but originel est de développer une équipe football, constituée de robots humanoïdes, capable de se confronter à une équipe humaine d'ici 2050. Globalement il s'agit de faire progresser la recherche en robotique dans un cadre compétitif afin de tirer le meilleur de chaque équipe.

Depuis la RoboCup s'est étoffée en accueillant de nouvelles ligues :

- La RoboCup Rescue réalise des simulations de sauvetage, en environnement hostile, par des robots.
- La RoboCup Junior développe la robotique pour les plus jeunes
- La RoboCup @Work simule un atelier où les robots font place aux humains
- La RoboCup @Home développe des robots d'assistance à la personne
- La RoboCup Logistic cherche à optimiser les chaînes de production en faisant appel aux robots.

C'est dans cette dernière que l'ARPL participe.

2 La Robocup Logistics League

Comme expliqué dans l'introduction cette ligue est sponsorisée par la société FESTO premier fournisseur mondial en technologie d'automatisation. Dans ce cadre, ceux sont les Robotinos, produit par FESTO, qui sont utilisés dans cette ligue.

Bien que ce soit une ligue récente (2012), elle a rapidement évolué. A ses débuts, elle n'était qu'une simulation de production avec des modifications représentées par des données numériques. Ce n'est qu'en 2015 que de véritables machines ont rejoins le terrain de 6 x 12 m.

Ce changement conséquent a apporté une grande quantité de challenges techniques d'un seul coup, ce qui explique le faible nombre de participants à l'Open German de 2015.

Ainsi dans cette usine vont s'affronter deux équipes de trois robots, ayant chacune six machines de production attitrées. Les robots d'équipes adverses ne peuvent pas communiquer entre eux, ainsi il y a de nombreuses variables inconnues sur le terrain. Chaque matchs se déroule en trois phases :

la phase préparatoire de cinq minutes pour paramétrer et mettre en place les robots, la phase d'exploration de quatre minutes pour repérer la position des machines et leur type (chaque machines à une tâche spécifique, à l'image d'une usine), et la phase de production de quinze minutes durant laquelle un arbitre informatique demande une production de pièces spécifiques, qui doit être optimisée par les robots.

3 Interface avec ROS:Robot Operating System

ROS fournit un environnement de travail qui peut être implémenté sur les robots afin de faciliter le développement de projets en équipe, en amenant des librairies, des outils et une architecture adaptable sur de nombreuses plate-formes. Cependant c'est en environnement prévu pour être implémenté sur les robots, ainsi il modifie le modèle de communication du robot avec l'extérieur.

Dans cette mesure, il est nécessaire d'avoir une interface spécifique à cette architecture, et en tant qu'environnement relativement récent, il n'en existe que très peu.

Cet environnement ne facilite pas la tâche de la phase préparatoire lors de la compétition, et il n'existe pas d'interface adaptée à nos besoins : gestion d'une équipe de robots en constante évolution.

Dans ce cadre une interface spécifique doit être développée, elle se voudra intuitive et ergonomique pour gagner du temps, générique pour palier à tout modification sur le robot et l'exporter dans d'autre projets. De plus elle pourra gérer plusieurs robots simultanément de manière sécurisé afin de respecter les normes de sécurité déjà mises en places dans le cadre de la compétition.

4 Cahier des charges

Pour résumé ce qui a était évoqué précédemment, l'interface devra respecter les critères suivants :

1. Ergonomique et intuitive
2. Gestion de plusieurs robots
3. Communication sécurisée
4. Généricité
5. Flexibilité

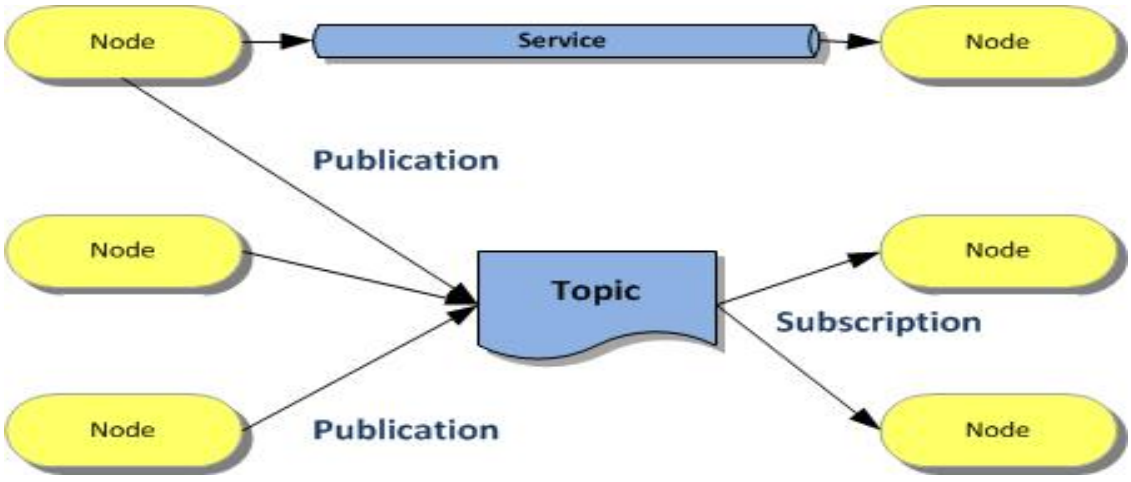
Présentation des outils utilisés dans le cadre du projet

1 ROS

En tant qu'environnement de développement, une présentation de son fonctionnement s'impose.

Le fonctionnement de ROS s'articule sur les éléments clés suivants : les nœuds (node), les topics et les services, ainsi que des variables « d'environnement » : les paramètres.

Voici un schéma synthétique de le communication entre ces éléments :



Les noeuds échangent entre eux via un système de topics, quand un nœud publie sur un topic il est appelé « publisher » et quand un nœud écoute un topic il est appelé « subscriber ». Un même nœud peut être subsciber et publisher.

Quand un nœud à besoin d'une information particulière, ou d'une modification singulière, il peut faire appel à un service fourni par un autre nœud.

Il existe de nombreuses distributions de ROS disponibles mais afin de pouvoir travailler sur un environnement stable avec les derniers outils, nous travaillons essentiellement sur ROS indigo compatible avec Ubuntu 14.04 LTS.

ROS présente aussi un grand nombre d'outils de diagnostique, débogage et d'affichage graphique interne.

2 Librairies ROS : Rosbridge-suite et RosbridgeServer

Parmis toutes les libraires de ROS, il en existe déjà une qui a été développée dans le cadre d'une communication avec un système externe n'utilisant pas directement l'architecture ROS. Celle-ci se trouve groupée au sein d'un ensemble de libraire regroupées dans rosbridge_suite :

https://github.com/RobotWebTools/rosbridge_suite.

Rosbridge_suite regroupe les libraires suivantes :

1. rosbridge
2. rosbridge_library
3. rosbridge_server
4. rosapi

Ensemble, elles fournissent une API pour communiquer avec ROS au travers de messages en JSON et la création d'un serveur web-socket côté robot.

Comme son nom l'indique, c'est rosbridge_server qui va crée celui-ci, avec la possibilité de choisir le port et le niveau de sécurité de la communication.

3 Librairies web: roslibjs

Parmi les librairies développé pour ROS mais pour un système extérieur, il existe la librairie roslibjs. Comme son nom l'indique celle-ci a été développé en javascript et fourni les outils nécessaire pour travailler avec le groupe de librairies cité précédemment.

Son fonctionnement repose sur la création d'une variable local globale représentant le système ros à l'extérieur. Ainsi elle fournit les outils pour récupérer et transmettre les informations via des messages JSON.

4 Langages web : JavaScript, Html, css

Étant donné l'existence d'une librairie en JavaScript, une interface web semble tout à fait approprié.

Cela permet de s'assurer de la flexibilité et du caractère portable de l'interface. De plus il existe de nombreuse autre libraires web compatibles avec ROS, et d'autres destinées au design d'interface :

https://github.com/RobotWebTools, https://jquery.com/.

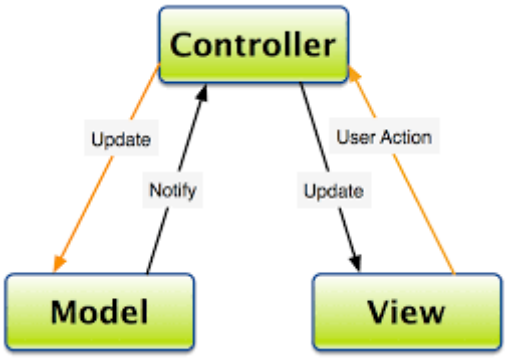
Celles-ci s'inscrivent dans l'architecture des pages web avec le langage HTML et les feuilles de style (CSS « Cascade Style Sheet »).

De plus, la grande quantité de libraires web compatible avec ROS proposées permettra d'ajouter de nombreuse fonctionnalités à l'interface pour des applications ciblées: retour flux vidéo, positionnement sur une carte etc. .

Présentation de l'interface

1 Architecture interne

Afin de pouvoir se retrouver facilement, j'ai opter pour une architecture de type MVC (Model View Controller).



Afin de coller à ce modèle, la partie « view » est représentée par notre page Html. C'est elle que voit l'utilisateur.

En arrière plan, pour mettre à jour, cette page, en l'absence de serveur d'hébergement, tout doit être réalisé côté client. C'est pourquoi des scripts JavaScript vont faire office de « controller ».

Ceux sont ses scripts qui vont amener les informations ROS à la page et vice-versa, lire et transmettre les commandes de l'utilisateur.

Finalement, les scripts directement liés à roslibjs sont intégrés dans la partie « model » étant donné qu'ils représentent le robot.

2 La partie « view »

Etant donné que c'est cette partie qui représente l'interface côté utilisateur, le caractère ergonomique de l'interface est quasiment entièrement lié à cette page. Tout les éléments présents de manière permanente ou en tant que références y sont présent. Le langage Html combiné au CSS permet de gérer la page tel un éditeur de texte: taille, placement, couleur, polices et davantage. Le langage Html est aussi nativement développé pour interagir avec avec l'utilisateur avec des champs de sélection, saisie textuelle, et d'affichage. Le CSS permet une gestion gloable de la page en terme de design. Par exemple, voici un extrait du modèle suivi par les onglets :

```
.tabs { border: 1px solid #aaaaaa; }

.tab-links
{
  background: grey;
  margin-top: 0px;
  padding-left: 1px;
}
```

```
.tab-links:after
{
   display:block;
   clear:both;
   content:'';
}

.tab-links li
{
   margin:0px 1px;
   float:left;
   list-style:none;
}
```

et la page principale scindée en deux groupes :

```
#container{width:100%;height: 100%;}
#Rtabs {overflow: auto; }
#Ltabs
{
        width: 30%;
        height: 100%;
        resize: both;
        float: left;
        overflow:
}
```

De manière globale, le bandeau supérieur de la page, les parties gauches et droites séparées en onglets et le modèle basique des onglets sont stockés dans cette page, confère extrait page html en annexe.

3 La partie « controller »

Cette partie représente une grande quantité de travail puisque c'est ici qu'est réalisé le traitement des information avant l'envoi vers le robot et celui des informations en provenance du robot.

Concrètement, c'est à cette partie qu'on doit le caractère générique de cette interface .

Afin de pouvoir gérer de multiples connections, la variable globale de roslibjs est dupliquée, de telle sorte que chaque robot (physique) ai la sienne. Ainsi lorsque l'on change d'onglet robot, la variable cible est modifiée.

En communication avec la partie « model », elle traite les informations en provenance du robot afin de pouvoir les afficher de manière compréhensible et fournir une interactivité directe entre l'interface et le robot. Elle récupère les topics, service et paramètres disponible pour fournir une autocomplétion sur champs de saisie de texte de l'utilisateur.

Lors de l'affichage ou de l'envoi d'un message, des algorithmes analyse la structure du message afin d'obtenir les champs d'affichage ou de remplissage par élément porteur de sens, et ceci tout en conservant les types de référence (chaînes de caractères, entiers etc.).

Globalement, il y a deux fichiers, pour la connexion et la gestion unitaire des robots (onglets de droite), avec un onglet s'ajoutant pour chaque robot ajouté, et chaque onglet proposant les même fonctions mais étant lié à un robot.

Et pour la partie de gauche, il y a un fichier par onglet.

Vous trouverez en annexe l'extrait d'un algorithme lisant le formulaire généré automatiquement pour intégrer les saisies de l'utilisateur dans un objet compréhensible pour roslibjs.

4 La partie « model »

C'est dans cette partie qu'est directement utilisé roslibjs. Les fonctions de la librairies ne sont pas utilisable tel quel pour être générique, la plupart des exemples d'utilisation transmette un message fixe à la fois.

Les script de cette partie permettent de récupérer les informations d'architecture de message, puis de flux de message pour l'onglet à gauche par exemple. Mais aussi la gestion des paramètres, des services et topics pour chaque robot.

Parfois, pour ne pas créer de fonctions inutiles, roslibjs est utilisé directement.

En annexe vous trouvez des exemples de fonctions utilisées.

5 Élément extérieur: connexion sécurisée

Sécurisation par certificat au sein du serveur et donc connexion ssl. La connexion est bien cryptée mais il est possible de se connecter en connaissant le port et en acceptant l'auto certification du serveur.

Exemple de commandes utilisées pour générer la clé:

openssl genrsa 1024 > servwiki.key

puis le certificat :

openssl req -new -key servwiki.key > servwiki.csr

En annexe vous trouverez la configuration côté serveur utilisant deux ports simultanément, un libre, et un sécurisé.

Conclusion

Bien qu'une interface soit la liaison entre deux éléments, ce projet de développement a essentiellement été orienté côté client. Cependant, c'est ce côté qui nous importe plus dans le cadre de la compétition.

L'ergonomie de l'interface n'a pas été définie au hasard, bien au contraire, elle est le fruit du mûre réflexion avec les membres de l'équipe que je remercie encore. De plus l'utilisation d'un gestionnaire de version : GitHub avec l'équipe est aussi un expérience enrichissante.

De plus d'un point de vue technique, ce projet a requit aussi bien des connaissance d'algorithmique, que de réseau ou de conception logiciel à objet sans oublier une petite partie de robotique.

La possibilité de pouvoir tester l'interface en local avec un simulateur est aussi un vrai plus pour le développement. Sans oublier le local de l'ARPL regroupant tout le matériel nécessaire au même endroit.

En terme d'aboutissement, le projet a été mené à bien vis à vis du cahier des charges et dans les temps. L'utilisation de librairies et technologies différentes de celles enseignées en cours est aussi un vrai plus en terme d'expérience, aussi bien technique, que de gestion de projet: il n'est pas aisé de créer et suivre un calendrier prévisionnel avec des techniques que l'on maîtrise peu.

Je compte poursuivre ma participation aux prochaines RoboCup avec l'équipe et dans ce cadre ajouter d'autre fonctionnalités à l'interface, comme évoqué dans ce rapport. De plus, il serait intéressant, de l'amener à un niveau standard plus développé afin de peut être l'utiliser dans un autre cadre ou la partager.

Annexe

Extrait page html

```
<header style="margin-top: 0px; margin-bottom: 10px;">  <!-- options disponibles en permanences-->
                <form method="post" action="#">
                        <input type="text" size="10" value="127.0.0.1:9090" id="ip">
                        <input type="text" size="10" value="robotino" id="name">
                        <button value="connecter" id="connecter" type="button"
onclick="addRobot(document.getElementById('secure').checked);">Connection</button>
                        <input type="checkbox" id="secure" checked="true">Connexion sécurisée
                        <input type="checkbox" id="save">Sauvegarde
                </form>
        </header>
        <div id="example" style="display: none;">
                <legend>Echange</legend>
                <fieldset style="height:150px; overflow-y: scroll;" >
                        <legend> Conversation </legend>
                        <div id="conversation"></div>
                </fieldset>

                <form name="echange" action="#" method="post">

                        <SELECT id="Liste" onChange="Lien_action()">
                        <option value="" selected="selected">Choisir une Action
                        <option value="topicPublisher">Publish a topic
                        <option value="topicSubscriber">Subscribe to a topic
                        <option value="serviceCall">Call a service
                        <option value="paramSetGet">Set and get Param
                        <option value="test">Test
                        </SELECT>
                        <br>
                        <div class="ui-widget">
                                <input type="text" id="echange1" value="" placeholder="" size="25">
                                <br>
                                <input type="text" id="echange2" value="" placeholder="" size="25">
                                <br>
                                <br>
                        </div>

                        <!--<textarea name="message" id="echange3" rows=4 cols=40></textarea>
                        <br>-->
                        <button onclick="lecture();" type="button" id="envoyer" value="envoyer"
title="Envoyer">Envoyer</button>
                        <button onclick="function_test();" type="button" id="test" value="test"
title="Test">Test</button>
                </form>
                <div id="echange_interactif">
                </div>
        </div>
```

Configuration serveur

```
<launch>
  <arg name="port" default="9091" />
  <arg name="port_ssl" default="9090" />
  <arg name="address" default="" />
  <arg name="ssl" default="false" />
  <arg name="certfile" default="/home/vianney3003/web-gui/private/server.crt"/>
  <arg name="keyfile" default="/home/vianney3003/web-gui/private/server.key" />
  <arg name="authenticate" default="false" />

<!--   <group if="$(arg ssl)">
    <node name="rosbridge_websocket_ssl" pkg="rosbridge_server" type="rosbridge_websocket" output="screen">
      <param name="certfile" value="$(arg certfile)" />
      <param name="keyfile" value="$(arg keyfile)" />
      <param name="authenticate" value="$(arg authenticate)" />
      <param name="port" value="$(arg port)"/>
      <param name="address" value="$(arg address)"/>
    </node>
  </group>
  <group unless="$(arg ssl)">
    <node name="rosbridge_websocket" pkg="rosbridge_server" type="rosbridge_websocket" output="screen">
      <param name="authenticate" value="$(arg authenticate)" />
      <param name="port" value="$(arg port)"/>
      <param name="address" value="$(arg address)"/>
    </node>
  </group> -->


  <node name="rosbridge_websocket_ssl" pkg="rosbridge_server" type="rosbridge_websocket" output="screen">
    <param name="certfile" value="$(arg certfile)" />
    <param name="keyfile" value="$(arg keyfile)" />
    <param name="authenticate" value="$(arg authenticate)" />
    <param name="port" value="$(arg port_ssl)"/>
    <param name="address" value="$(arg address)"/>
  </node>

  <node name="rosbridge_websocket" pkg="rosbridge_server" type="rosbridge_websocket" output="screen">
    <param name="authenticate" value="$(arg authenticate)" />
    <param name="port" value="$(arg port)"/>
    <param name="address" value="$(arg address)"/>
  </node>


  <node name="rosapi" pkg="rosapi" type="rosapi_node" />
</launch>
```

Communication avec ROS

```
function subscriber(TopicName, MessageType, Ros)
{
	var topic = new ROSLIB.Topic({
		ros :Ros,
		name : TopicName,
		messageType : MessageType
	});


	topic.subscribe(function(message)
	{
		console.log('received message on ' + topic.name + ': ' + message);
		console.log(message);
		var data='';
		for (var key in message)
		{

		data += key +' ' + message[key]+'; ';
		}
		converse(data);
		topic.unsubscribe();
	});
}

function getSrvRequestDetails(srvType, Ros, callback)
{
	var serviceRequestInfo = new ROSLIB.Service(
	{
		ros : Ros,
		name : '/rosapi/service_request_details',
		type : 'rosapi/ServiceRequestDetails'
	});

	var request = new ROSLIB.ServiceRequest(
	{
		type : srvType
	});

	serviceRequestInfo.callService(request, function(result)
	{
		console.log(result);
		console.log(result.typedefs)
		if (typeof callback === 'function')
		{
			callback(result.typedefs);
		};
	});

}
```

Algorithme permettant de générer un objet à partir d'informations saisies

```
function genFromForm(typedef)
{
	var navigateur = new Array;
	var i = 1,j = 0;
	var floatType = ["float64","float32","float16"];
	var intType = ["uint8","uint16","uint32","uint64"];
	if (rosElement.querySelector('#i'+i).innerHTML == '' && i == 1) //seulement une ligne d'input
	{
		while (rosElement.querySelector('#i'+i) != null)
		{
			if ((rosElement.querySelector('#i'+i).placeholder.indexOf("float") >= 0) &&
floatType.indexOf(rosElement.querySelector('#i'+i).placeholder) != -1)

	typedef[rosElement.querySelector('#i'+i).name]=parseFloat(rosElement.querySelector('#i'+i).value);
			if ((rosElement.querySelector('#i'+i).placeholder.indexOf("int") >= 0) &&
intType.indexOf(rosElement.querySelector('#i'+i).placeholder) != -1)

	typedef[rosElement.querySelector('#i'+i).name]=parseInt(rosElement.querySelector('#i'+i).value);
			if (rosElement.querySelector('#i'+i).placeholder == "string")

	typedef[rosElement.querySelector('#i'+i).name]=rosElement.querySelector('#i'+i).value.toString();
			i++
		}
	}
	else //objet complexe
	{
		while (rosElement.querySelector('#i'+i) != null)
		{
			if  (rosElement.querySelector('#i'+i).innerHTML != '')
			{
				if (i != 1)
					j++;
				navigateur[j] = rosElement.querySelector('#i'+i).innerHTML;
			}
			else
			{
				if ((rosElement.querySelector('#i'+i).placeholder.indexOf("float") >= 0) &&
floatType.indexOf(rosElement.querySelector('#i'+i).placeholder) != -1)
					typedef[navigateur[j]]
[rosElement.querySelector('#i'+i).name]=parseFloat(rosElement.querySelector('#i'+i).value);
				if ((rosElement.querySelector('#i'+i).placeholder.indexOf("int") >= 0) &&
intType.indexOf(rosElement.querySelector('#i'+i).placeholder) != -1)
					typedef[navigateur[j]]
[rosElement.querySelector('#i'+i).name]=parseInt(rosElement.querySelector('#i'+i).value);
				if (rosElement.querySelector('#i'+i).placeholder == "string")
					typedef[navigateur[j]]
[rosElement.querySelector('#i'+i).name]=rosElement.querySelector('#i'+i).value.toString();
			}
			i++;
		}
	}
	return typedef;
}
```