Projet de Bureau d'étude

Yousre Ouali & Hicheme BEN GAIED
HE2B: ISIB
54667@etu.he2b.be & 58930@etu.he2b.be

Résumé

L'objectif de notre projet est de programmer le robot KUKA afin qu'il puisse taper sur un clavier. On a décidé de faire un compte Twitter au robot et lui faire Twitter des phrases. Ces phrases auront pour sujet Pokemon. Il nous dira quelles sont les évolutions des Pokemon.

Mots-clefs : Robot, KUKA, Clavier, Twitter, Pokemon, Souris

I. Introduction

Notre projet consiste à programmer un robot KUKA de telle sorte qu'il puisse écrire sur un clavier seul.

I. KUKA

Dans le cadre de mon projet d'étude, on a développé un système permettant à un bras mécanique KUKA de taper des choses sur un clavier. L'automatisation des tâches est devenue essentielle dans de nombreux domaines, et l'utilisation de bras mécaniques offre de vastes possibilités pour améliorer l'efficacité et la précision des processus.

Kuka est avant tout une entreprise de robotique. C'est une société mondiale d'automatisation. Le siège de la société se trouve en Allemagne.

II. Description des différentes parties du projet

Lors de ce projet, nous sommes passer par plusieurs étapes. Voici une description brève des étapes ;

- Maintenir une connexion avec le robot.
- Configurer ses entrés pour qu'il puisse recevoir des coordonnées.
- Programmer en KRL afin qu'il puisse utilisé ses entrés et procéder à des mouvements.
- Bouger le robot avec les coordonnées envoyées.
- Prendre les coordonnées des touches du clavier.
- Bouger le robot pour qu'il puisse taper sur le clavier juste en lui donnant la phrase à taper.
- Initialiser les étapes à suivre pour que le robot sache ou appuyer pour réaliser un post.

III. Matériel et Méthode

Les outils utilisés sont les suivants ;

- Un robot KUKA
- Un ordinateur
- Un clavier mécanique
- Un support pour le clavier

En ce qui concerne le language utilisé lors de ce projet. On a dû utiliser deux languages de programmation, **Python** et le **KRL**.

Python a été utilisé pour instancié le serveur. Ce serveur permet de maintenir la connexion, recevoir/envoyer des coordonnées, vérifier des coordonnées et séquencer la phrase que le robot doit taper. Les informations que le serveur envoie au robot est sous format **XML**.

Le KRL permet de se connecter au serveur et d'utiliser les informations que le serveur lui envoie. Pour que le connexion se fasse, le programme a besoin d'information tels que l'adresse ip et le port du serveur. Ces informatios se retrouve dans un fichier XML. Ce fichier possède aussi d'autres informations tels que les données en entrés et en sortie du robot.

Et en ce qui concerne le support de clavier, ce support est obligatoire en vue de respecter la zone de travail du robot. En effet, le robot possède une zone de travail. Lorsque le robot est hors de cette zone, le robot est inutillisable. Cette zone permet d'avoir une sécurité. Elle permet en autre d'éviter que le robot soit en conatct avec le plan de travail par exemple.

Remarque, la phrase que le robot devra taper ne doit pas être longue. Si la phrase est trop longue à écrire, le temps pour que le robot tape la phrase est prolongé.

I. Connexion

En ce qui concerne la connexion. Le robot est connecté en **VLAN**. Donc, il n'est pas directement lié au réseau. Pour pouvoir communiquer avec celui-ci, on est obligé de se connecte au VLAN.

Lorsque la communication entre le serveur et le robot est assurés, on peut passer à l'étape suivante qui consiste à créer un fichier XML. Ce fichier XML contiendra toute les informations afin d'assurer la connexion. Comme par exemple, l'adresse ip du serveur, le port que le serveur utilise et le protocol utilisé pour communiquer.

Et enfin, lorsque le fichier XML est bien configuré, on peut passer au programme KRL. On a crée un objet **RSI** de type **ST_ETHERNET**. Cette objet spécifique au language **KRL** permettra d'initialisé la connexion. Lorsque la connexion a été établie, le robot commence par envoyé le premier fichier XML au serveur. Ce premier fichier contiendra des informations comme les coordonnées, mais aussi un **IPOC**. Cette valeur est composée de chiffre. Elle devra être renvoyé au robot selon un temps impartie. Si cette valeur n'est pas renvoyé dans les plus bréves délais, alors la connexion s'interromt.

II. Configuration du fichier XML

Comme expliqué précédement, le fichier XML permet de configuré la connexion du robot au serveur. Toutefois, ce fichier contient d'autre informations. Voyons voir de plus près l'architecture de ce fichier.

La balise racine de ce fichier se nomme "**ROOT**". Dans la balise racine se trouve 3 autres balises. Il y a une balise qui se nomme "**CONFIG**", une autre "**SEND**" et la dernière s'appelle "**RECEIVE**".

La balise **CONFIG** va permettre de configurer les paramètres de communication. On va y retrouver 6 autres balises ;

- "**IP_NUMBER**" contenant l'addresse IP du serveur.
- "**PORT**" permettant d'avoir le port utilisé par le serveur.
- "**PROTOCOL**" qui défini le protocole utilisé pour la communication. On a deux types de protocoles utilisable, on a le droit d'utiliser le protocole **TCP** ou **UDP**.
- "**SENDTYPE**" est l'identifieur du système externe. Cette valeur va être identifier pour chaques paquets reçu au robot.
- "**PROTOCOLLENGTH**" est une balise permettant d'activier la transmission de la longueur de byte du protocole. Elle n'a que deux valeurs possibles ("**ON**" ou "**OFF**").
- Et enfin, la balise "**ONLYSEND**" permet de définir la direction des échanges de donnée. Elle peut être mis à "**TRUE**", ceci permet au robot d'envoyer des données mais pas d'en recevoir. Tandis que, lorsqu'il est mis à "**FALSE**", alors le système peut envoyer et recevoir des données.

Concernant la balise **PROTOCOLLENGTH**, il a été initialisé à **OFF** et la balise **ONLYSEND** a été initialisé à **FALSE**.

Ensuite, la balise **SEND** permet de définir ce que le robot peut envoyer. Cette balise contient une sous-balise "**ELEMENTS**". Cette balise conteint aussi d'autres balises "**ELEMENT**". La balise **ELEMENT** a des attributs tels que ;

- **TAG** est le nom du tag qui sera généré par le robot.
- **TYPE** est le type de donnée.
- **INDX** est le numéro de sorite de l'objet.
- Et enfin, **UNIT** est l'unité utilisé pour la donnée.

Pour ce projet, j'ai juste besoin des coordonnées du robot. Le système possède déjà un objet préconfiguré qui se nomme "**DEF_RIst**". C'est dans l'attribut **TAG** où on précise le nom de cette objet. Le type de cette objet est un **DOUBLE**, le numéro de sortie est **INTERNAL** et l'unité est **0**.

Remarque, il existe plein d'autres mots-clés comme **DEF_RSol** (envoie les commandes de positions cartésienne), ou le **DEF_MACur** (envoie les courants moteur du robot de l'axe A1 à A6).

Et enfin, la balise **RECEIVE** est la balise qui définie la sortie du robot. Cette balise à la même architecture que la balise **SEND**. Toutefois, la balise **ELEMENT** contient un attribut en plus, c'est le **HOLDON**. Ce nouveau attribut va définir le comportement de l'objet reçu au robot lorsque celle-ci à une erreur. Elle a deux valeurs possibles ;

- **0** permet de réinitialisé la veleurs reçu.
- **1** permet de maintenir l'ancienne valeur valide alors que la nouvelle valeur a une erreur.

Dans ce projet, tous les eléments ont un **HOLDON** à **1** afin d'assurer une localisation toujour valide.

III. Programmation en KRL

Passons maintenant dans le language KRL, on va définir les différents principes et outils utilisés dans le programme.

Au début du programme, on commence d'abord par déclaré les variables utiles. Une variable de type **RSIERR** est déclaré. Cette variable permet, en autre, de récupérer la réponse d'une commande. Des variables de type **INT** seront utilisées afin de stocker des entiers. Il y aura un entier (nommé **hEthernet**) contenant l'identifiant de l'objet permettant l'accés à l'objet **RSI**. Cette valeur est automatiquement assigné par le système lorsque l'objet RSI est créer. D'autres entiers seront utilisés pour identifié l'objet RSI. Et enfin, une variable de type **FRAME** sera déclaré. C'est une variable permettant de faire la liaison avec le capteur de coordonnée du système. Elle permet notemment de changer les coordonnées selon des nouvelles coordonnées reçu par le robot.

Lorsque les déclarations de variables sont terminées. On peut alors passer à la configuration de l'objet **RSIERR** avec des méthodes mises à dispositions.

La communication entre le controlleur du robot et un système externe est implémenté en utilisant l'objet **RSI ST_ETHERNET**. Cette objet doit être crée et configuré dans le programme KRL afin de établir la communication. Pour activer la communication entre ces systèmes, l'utilisateur doit créer et configurer le fichier XML. Ce fichier de configuration est spécifié et charger lorsque l'objet RSI ST_ETHERNET est crée. Cette objet prend trois paramètres.

Le premier paramètre est l'identifiant de l'objet permettant l'accés à l'objet **RSI**. Dans notre cas, cette variable est appellé **hEthernet**.

Le deuxième paramètre est un entier correspondant au nombre de containeur dans lequel l'objet RSI est crée. Nous ne voulons pas créer de containeur, on va donc précisé qu'aucun

containeur doit être crée.

Et enfin, le troisème paramétre est le chemin d'accés vers le fichier de configuration XML.

L'objet RSI ST_ETHERNET peut être configurer à l'aide de la commande **ST_SETPARAM**. La seule configuration faite concerne le nombre maximum de paquet de donnée pouvant arrivé en retard au controlleur du robot. Ce paramètre se nomme **eERXmaxLatePackages** et on a définie ce nombre maximum à 500. Ce qui signifie que 500 est le nombre maximum de paquet de donnée qui peut être en retard.

Lorsque la partie concernant la connexion est terminé, on peut commencer à mapper les entrés de données du robot. Pour réaliser cela, on utilise l'objet **RSI ST_MAP2SEN_PREA**. Cette objet permet donc de rendre disponible la lecture de donnée que le système externe envoie. Il prend cinq paramètres ;

- Le premier paramètre est l'identifiant de l'objet. C'est donc un entier. Dans le programme KRL, il sera nommé **hMap2Prea**.
- Le deuxième paramètre est le numéro du containeur de l'objet RSI, dans notre cas, c'est le containeur 0.
- Le troisème paramètre est l'identifiant de l'objet permettant l'accés à l'objet **RSI** (hEthernet).
- Ensuite, le quatrième paramètre est l'index de la source de signal. Ceci est directement lié à la configuration du fichier XML, dans la balise **RECEIVE**.
- Et enfin, le cinquième paramètre est l'indice de la variable de système **$SEN_PREA[]**.

Remarque importante, les variables seront accessible avec la variables du système **$SEN_PREA[X]**. Où le terme "X" est l'indice qu'on utilise pour accéder à la variable reçu.

Pour la sécurité du matériel, on active un cycle de correction. Ce cycle permet de corrigé les mouvements selon le système de référence qu'on utilise. On utilise le système **#WORLD**. Pour activé cette correction, on utilise la commande **ST_ON1(#WORLD,1)**.

On peut enfin passer dans la création d'une boucle permettant de récupérer les nouvelles coordonnées envoyé par le système extenre. Dans cette boucle, on utilise la variable de type **FRAME**. Cette variable va contenir les nouvelles coordonnées à l'aide de la variable **$SEN_PREA[X]**. Lorsque les coordonnées sont initialisées, alors on peut procéder à un mouvement de type **PTP** (Point To Point). Ce type de mouvement point par point ne permet pas d'avoir un mouvement linéaire. Il permet juste de passer à un point dans l'espace qu'on définie avec une coordonnée.

IV. Mouvement

Dans cette partie, nous allons discuter des différents types de mouvements effectués par le bras Kuka afin de répondre aux différents besoins de notre projet. Les mouvements que nous allons aborder vont permettre d'écrire sur le clavier et de publier le tweet.

IV.1 Mouvement clavier

Le mouvement du clavier a été décomposé en 3 parties. La première étape consiste à se positionner au-dessus de la touche à écrire. Le bras Kuka se déplace en X et Y uniquement pour cette phase. Une fois positionné au-dessus de la touche, le bras effectue un déplacement en Z pour appuyer sur la touche. Après avoir pressé la touche, le bras effectue à nouveau un déplacement en Z pour se retirer de la touche. Ce cycle est répété jusqu'à ce que le bras ait fini d'écrire la phrase souhaitée. Les coordonnées de chaque touche sont stockées dans un dictionnaire appelé "Dico" dans le fichier MovementManager.py. Ces coordonnées sont transmises au robot en fonction de la touche à écrire. Elles permettent également d'effectuer plusieurs vérifications en comparant la position à atteindre et la position actuelle du bras :

1. Vérifier que le robot se trouve bien au-dessus de la touche concernée.

2. Vérifier que le robot a bien appuyé sur la touche du clavier.
3. Vérifier que le robot s'est bien relevé après l'appui.

Ces comparaisons permettent de garantir la précision et l'exactitude des mouvements à effectuer par le bras Kuka lors de l'écriture sur le clavier.

IV.2 Soumission du tweet

Une fois la phrase entièrement écrite, le robot doit maintenant soumettre le tweet. Pour cela, nous utilisons la touche de tabulation huit fois afin de sélectionner le bouton "Tweet". Une fois ce bouton sélectionné, nous ordonnons au robot de se diriger vers la touche "Enter" du clavier et d'appuyer dessus pour soumettre notre tweet. Les coordonnées de la touche "Enter" et "Tabulation" sont également stockées dans un dictionnaire appelé "MousePos", qui se trouve dans le fichier MovementManager.py. Les mêmes vérifications que celles effectuées pour l'écriture sur le clavier sont appliquées pour soumettre le tweet. Une fois notre tweet soumis, le robot peut écrire un autre message s'il le souhaite en répétant le même schéma.

V. Twitter

Lors du lancement du programme, le robot va directement appuyer sur la touche "N". Cette touche est un raccourci proposé par Twitter pour ouvrir rapidement et facilement la fenêtre d'écriture des tweets. Les coordonnées permettant d'utiliser cette option sont stockées dans le dictionnaire "MousePos" sous le nom de "new", situé dans le fichier MovementManager.py. Une fois la fenêtre ouverte, le robot va commencer à rédiger son tweet, puis le soumettre.

IV. Problème rencontré

Durant notre projet, nous avons rencontré un problème lié à l'utilisation de la souris par le bras. En effet, notre objectif était d'utiliser la souris pour déplacer le curseur vers le bouton "Tweet" afin de soumettre le tweet rédigé par le robot, au lieu d'utiliser les huit tabulations. Pour cela, nous avons développé un boîtier adapté à la tête du robot et implémenté le code nécessaire pour cette fonctionnalité. Lors de nos tests, cela fonctionnait très bien, mais lorsque nous sommes passés en mode T2, qui permet au robot d'aller plus vite, un problème majeur et évident est apparu. La distance parcourue par le curseur variait en fonction de la vitesse. Malheureusement, nous n'avions pas eu la possibilité de calibrer la souris avec le mode T2, car nous ne disposions pas de la clé nécessaire pour activer ce mode. Nous pensons toutefois qu'il est possible de résoudre ce problème en ajoutant du code dans le fichier contenant les instructions KRL, de manière à imposer une vitesse au robot lors de l'utilisation de la souris.

Cependant, par manque de temps, nous avons finalement opté pour la méthode des tabulations et de la touche Enter afin de publier nos tweets.

V. Résultat final

Le résultat est que le robot KUKA tape sur les touches du clavier selon la phrase qu'on fournie au sysème externe. Dans notre cas, on a fait en sorte que le serveur (système externe) fasse une requête à une API. Cette API donne des informations sur les évolutions des **POKEMON**.

Le robot a aussi la possibilité de bouger la souris pour soumettre un Tweet. Ce type de mouvement ne peut se faire qu'au niveau T1 du bras mécanique.

VI. Améliorations possibles

Les améliorations sont nombreuses comme ;

1. Équipé le robot d'une caméra, afin qu'il puisse reconnaître les lettres dans le clavier. Cela serait utile afin d'éviter d'écrire

toutes les coordonnées de chaque lettre du clavier.

2. Équipé le bout du robot avec un moteur qui viendait taper sur la touche du clavier en peu de temps. Ceci permettrai d'éviter les mouvements sur l'axe Z du robot. Et donc, la rédaction de la phrase serait plus rapide.
3. Générer des phrases avec une IA. Cette amélioration a été travailler mais sans succés. Le problème est qu'il faut payer pour avoir un quotas. Ce quotas est le nombre de requête qu'on peut envoyé à l'IA. Cette amélioration permettrai d'avoir une large gamme de sujet et des phrases différentes à chaque tweet.
4. Et enfin, équilibré le mouvement de la souris avec le niveau T2 du robot KUKA.

Références

[Kuka Roboter] 2009
KUKA System technology KUKA.RobotSensorInterface 2.3 / For KUKA System Software 5.4, 5.5, 5.6, 7.0