Rapport de Projet Scientifique et Technique : fabrication d'un bras robotique de pelleteuse.

Victor Mazoyer & Robin Menelot

INSA de Lyon Génie Mécanique

Résumé

Ce rapport présente le Project Scientifique et Techinique que nous avons mené à bien lors de notre second semestre de troisième année au sein du département Génie Mécanique à l'INSA de Lyon. Vous pourrez trouver dans ce rapport la synthèse de notre travail, ainsi que nos méthodes et nos résultats. Nous souhaitons remercier notre tuteur Nans Biboulet pour son soutient, nos coéquipiers qui travaillaient sur des projets similaires et avec qui nous avons su maintenir des relations d'entre-aide.

I. Introduction

L'objectif est de réaliser un bras de pelleteuse en utilisant un kit de Lego Mindstorms EV3. Le kit nous permet de réaliser rapidement des prototypes fonctionnels avec des composants interchangeables. Nous avions déjà utilisé ce kit pendant la semaine d'intégration de GM et en P2i mécatronique. Cependant nous ne pilotions pas les moteurs/-capteurs avec les même outils.

Pour la partie programmation l'objectif est de piloter entièrement le bras à travers un réseau sans fil. Et pour cela il a fallu utiliser et paramètrer un environnement de programmation avec lequel nous n'étions pas familier. Ce projet permet donc d'apprendre à utiliser le terminal Linux, la création d'architecture Réseau sous Python et enfin de gérer les moteurs du Kit Lego.

i. Cahier des charges

> Voici les instructions que nous avions avant de commencer notre projet :
> `Réaliser en lego ev3 une cabine d'engin de chantier type pelleteuse. Commander à travers un réseau sans les mouvements de l'engin: rotation horizontale, bras, godet. Vous commanderez l'engin en utilisant les capteurs de votre smartphone, un peu comme une manette de Wii, (ou à défaut d'un raspberry équipé d'une board de capteurs).`

II. Programmation

Contrairement à ce qu'il peut paraître, la grosse majorité de notre projet était en réalité de la programmation. Entre l'apprentissage des différents outils, la familiarisation du nouvel environnement informatique et l'écriture du code, notre temps de travail s'est concentré sur la partie numérique du projet. Les choix d'outils liées à notre projet se sont révélés être largement utilisés à travers le monde, et donc on avait accès à de nombreuses aides et ressources sur internet.

i. Architecture

i.1 Composants du système

Notre système se compose en quatres parties plus ou moins indépendantes : l'ordinateur, le Raspberry Pi, la brique Lego, les moteurs. Chaque composant joue un rôle particulier et déterminant dans le système. Il nous était imposé d'utiliser chaque composant.

L'ordinateur était un élément principal lors de notre travail, en effet, celui-ci nous permettait d'écrire nos programmes sur les autres composants via une connection SSH, faire nos recherche et piloter notre bras robotique. Par ailleurs, on a fait le choix d'utiliser Linux ; de naviguer dans nos fichiers et d'écrire nos programmes grâce au Terminal et à l'éditeur de texte Vim.

Le Raspberry Pi[1], est une carte électronique (de la taille d'une grosse carte bleu), qui se comporte comme un ordinateur. Dotée de port HDMI, USB, d'un émetteur/receveur WIFI, d'un accéléromètre piézoélectrique et bien d'autre encore, nous a servi comme serveur et de relais entre les autres composants, en plus d'être une manette tridimensionnelle pour notre robot.

La brique Lego, est le relais entre les informations numériques et les sorties mécaniques des moteurs. La briques à pour rôle de convertir les instructions codées en mouvements physiques des moteurs constituants notre robot.

i.2 Interaction des parties

Afin de transformer chaque composant indépendant en un système entier il faut les faire communiquer. Au vu de nos connaissances en informatique, on a opté pour un système de communication impliquant des sockets, et donc un système serveur/client.

1. On le référencera par 'RasPi' dans la suite du rapport

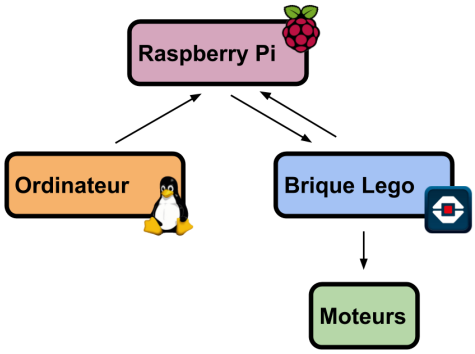

Figure 1 – *Diagramme des communications entre les composants*

La figure 1 synthétise le fonctionnement des communications entre nos différents composants. On peut voir que le RasPi est un élèment central au protocol, agissant en tant que serveur il devient le relais de chaque transit d'information. On remarque aussi que toutes connections représentées sur le diagramme sont dans la réalité des communications wifi locales, sauf la brique, qui est connectée aux moteurs par des cables élèctroniques. Notre sytème présente deux fonctionalités avec deux protocoles légèrement différents.

La première, comme indiquée dans le cahiers des charges, une télécomande tridimensionnelle qui pilote les différents moteurs en fonction de ses déplacements dans l'espace (façon manette de Wii). Ce protocole ne requiert par l'ordinateur ; un thread de communication python est ouvert entre le RasPi et la brique, on récupère les accélérations des capteurs piézoélectriques du RasPi avant des les analyser et les envoyer à la brique par wifi. Le fonctionement du système sera expliqué plus loin dans le rapport.

La seconde, un désir de notre part (notamment à cause de la faible précision de la prmière façon de fonctionner), était de pouvoir controler les moteurs grâce au clavier de l'ordinateur. On a simplement rajouté l'ordinateur à la communication, le RasPi communique

donc avec deux clients simultanement. Il est d'autant plus important d'établir un protocole de communication rigoureux afin d'établir les règles et l'ordre des informations échangées entre les composants.

ii. Code

Cette section a pour but d'expliquer les outils principaux utilisés lors de la programmation. Il faut savoir que le language utlisé pour la programmation de notre système est le PYTHON. Ce choix nous était d'une part imposé, mais s'est révélé judicieux de part la grande quantité de documentation et de LIBRARIES disponible sur internet.

ii.1 Terminal Linux et SSH

Notre environement de travail informatique peut se résumer au Terminal Linux et son éditeur de texte Vim. L'utilisation simple des commandes CD, LS, MAKEDIR ainsi que VIM et PYTHON3 nous on rapidement permis de prendre en main les outils pour débuter notre programmation.

La connection SSH était aussi une partie importante, puisque c'est grace à celle-ci que nous pouvions sans effort écrire des programmes sur les différents composants. Avec une simple ligne de code dans le Terminal on pouvait commencer à programmer sur les appareils.

ii.2 Socket

Les SOCKETS existent dans différents languages informatiques, dans notre cas la PYTHON. Elles sont au centre des systèmes de communication, nous ne nous sommes pas penchés vers les détails de leurs fonctionnements. En revanche, on savait qu'il fallait choisir judicieusement le serveur et les clients. Nous avons choisi le RasPi comme serveur pour trois raisons : c'est un appreil à la puissance de calcul rapide contrairement à la brique Lego ; il est présent dans les deux protocoles de communication, c'était donc plus simple pour nous ; finalement, il est au centre des transits d'information, c'est donc une minimisation du nombre d'échanges, et donc un gain de temps.

ii.3 Capteurs piézoélectrique

En utilisant les capteurs du RasPi et les libraries XLoBorg, on pu simplement extraire les accélérations (en multiple de $\vec{g}$, l'accélération terrestre) dans les directions $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ du repère associé au RasPi.

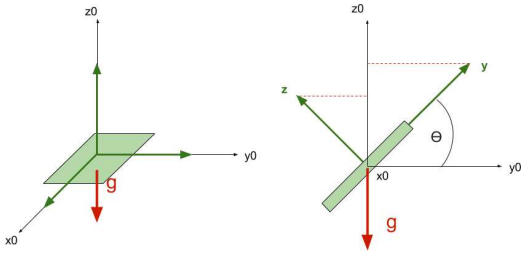

Figure 2 – *Schema qui reprérente le repère galiléen, le repère du RasPi et l'accélration terreste.*

Lorsque le RasPi est à plat les valeurs retournées sont $\begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}$ car c'est la projection de $\vec{g}$ sur le repère du RasPi. Maintenant, si on incline le RasPi d'un angle θ selon x_0, les projections deviennent :

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \cdot \vec{g} = \begin{cases} +0 \\ -g\sin\theta \\ -g\cos\theta \end{cases}$$

On peut donc voir que la direction dans laquelle l'accélération est nulle, est l'axe de rotation du RasPi. Tout ceci est bien joli, mais ne fonctionnent que pour les axes x et y car il est impossible d'exploiter ce phénomène afin d'avoir $\vec{z} \cdot \vec{g} = 0$ et des valeurs de projection non nulles dans les deux autres directions. Ceci veut dire qu'on a seulement deux degrés de liberté, donc qu'on ne peut controler que deux moteurs indépendement.

On a pensé à une solution afin de contourner le problème, mais elle semblait au-dessus de nos compétences. Il s'agit d'établir un changement de base du repère associé au RasPi

pour le transformé en "pyramide inversé", cela aurais la conséquence de parvenir à des projection de $\vec{g}$ non nulles dans toutes les directions du nouveau repère du RasPi.

ii.4 Getch

GETCH nous a servis à la lecture en continu des touches du clavier. Pour controler le robot à l'aide des flèches du clavier on devait mettre en place un système de KEY LISTENER non-bloquant et continu. Les libraries Getch on remplis cette fonction.

ii.5 Library EV3

Les lignes de codes présente sur la brique Lego sont majoritairement des instructions pour les moteurs. Ces lignes de code font appel à des fonctions disponibles dans des libraries en open-source sur internet. Voici un exmple d'une ligne de code qui fait appelà une fonction EV3 :

```
mB.run_timed(time_sp=600,
speed_sp=600)
```

Ici cette ligne de code ordonne au moteur B de tourner pendant 600 ms à vitesse 600 (échelle de puissance : [0 , 1400]).

III. MÉCANIQUE

Intro chapitre mécanique
construciton par essai et echec
bricolage
on a fait ça comme mais ça aurais pu être différent
introduire les legos

i. Cinématique du bras

plagia sur bras de pellteuse
on avait que 3 moteurs
image de scheme cinématique +commentaire

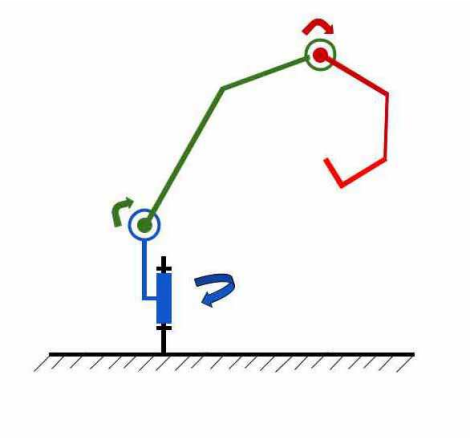

Figure 3 – *Schema cinématique du bras robotique.*

ii. Construction

parler vite fait des lego
parler des problème : elastique + vis sans fin (avec photo de la vis)

ii.1 Prototype

finir par photo du prototype final et parler de ce qu'on a réussi à faire concretement

RÉFÉRENCES

[1] EV3 PYTHON. Learn EV3 Python [en ligne]. Disponible sur : `https://sites.google.com/site/ev3python/` (22/03/17)

[2] PiBorg. Robotics add on boards for use with your Raspberry Pi [en ligne]. Disponible sur : `https://www.piborg.org/xloborg` (19/05/17)

[3] N. Biboulet. Démarage PST Lego [PDF] 2017. 7 pages.

IV. Annexes

i. Programmes sur ordinateur

i.1 clientPCv1.py

```
import socket
import time
from Getch import Getch,whatKeyIsPressed

if __name__ == '__main__':

        s = socket.socket()
        host = '192.168.42.1'
        port = 8087
        s.connect((host,port))

        while True:
                time.sleep(0.1)
                key = whatKeyIsPressed()
                s.send(key.encode())
                print(key)
                if key == 'q':
                        s.close()
                        break
}
```

i.2 Getch.py

```
class Getch:
        def __init__(self):
                import tty, sys

        def __call__(self):
                import sys, tty, termios
                fd = sys.stdin.fileno()
                old_settings = termios.tcgetattr(fd)
                try:
                        tty.setraw(sys.stdin.fileno())
                        ch = sys.stdin.read(1)
                finally:
                        termios.tcsetattr(fd, termios.TCSADRAIN, old_settings)
                return ch

def whatKeyIsPressed () :
        while True:
                init = Getch()()
                if init == 'q':
                        return 'q'
                        break
                if init == 'z':
                        return 'z'
                if init == 's':
                        return 's'
                elif init == '\033':
                        Getch()()
                        key = Getch()()
                        if key=='A':
                                return 'up'
                        if key=='B':
                                return 'down'
                        if key=='C':
                                return 'right'
                        if key=='D':
                                return 'left'
```