

Compte Rendu Démarche Experimental ***Robot mobile Thymio II***



Jassim El Akrouch
Yanis Benbachir

Sommaire

I-	Introduction.....
II-	Les outils de travail.....
	a) Le robot Thymio II.....
	b) Raspberry Pi 2.....
	c) Carte MicroSD
	d) Les câbles.....
	e) Camera
III-	Le projet.....
	a) Assemblage du robot.....
	b) La connexion avec l'ordinateur.....
	c) La connexion entre le robot et le raspberry.....
	d) Le Fonctionnement du robot
IV-	Conclusion.....
V-	ANNEXES.....

I- Introduction

Durant les trois travaux pratiques de démarche expérimental il nous a été demandé de programmer un robot Thymio II afin que ce robot puisse suivre une ligne noire sur fond blanc. Nous allons voir au long de ce compte rendu le matériel que nous avons utilisé et comment nous sommes arrivés à programmer le robot en langage Python.

II- Les outils de travail

a) Le robot Thymio 2

Le Thymio II est un robot développé en collaboration par l'École Polytechnique Fédérale de Lausanne (EPFL) et l'École Cantonale d'Art de Lausanne. Leur but est de fournir un robot éducatif à bas prix.

Le Thymio II est totalement open source, que ce soit au niveau logiciel ou matériel.

Le robot mobile Thymio 2 est une petite base roulante interactive, avec de nombreux capteurs, un logiciel de programmation graphique et un textuel de très bonne facture et de nombreux exemples et document. Le robot Thymio 2 a été conçu pour l'éducation

b) Raspberry Pi 2

La Raspberry pi est un nano-ordinateur de taille d'une carte crédit équipé d'un processeur Broadcom BCM2836, quatre cœurs ARMv7 à 900 MHz, accompagné de 1 Go de RAM. IL nous a permis de traiter l'image reçue de la caméra afin de calculer les vitesses des roues et les envoyer sous forme de consignes au robot. Le calcul est fait avec un programme codé en python.

c) Carte Mémoire MicroSD

Carte mémoire flash MicroSD HC classe 4 d'une capacité de 4Go.

La mémoire du Raspberry, contient le système d'exploitation RASPBIAN (basée sur DEBIAN JESSIE) et les programmes de python.

d) Les câbles

Alimentation : La Raspberry est branché à une batterie mobile (Power Bank) qui lui fournit 5 Volts et 2 Ampère. Elle facilite le déplacement du robot.

Câble RG45 : Le câble réseau RG45 sert à connecter la Raspberry au réseau interne via un switch.

e) Camera

Connecté au Raspberry par une nappe dans un port spéciale (CSI port), la caméra est capable de prendre des vidéos de 1080p/30fps ou encore 720p/60fps. Elle est placée au centre du robot. Le module de caméra Raspberry Pi peut être utilisé pour prendre la vidéo haute définition, ainsi que les photos. Il est facile à utiliser pour les débutants, dans notre projet la camera va nous servir à prendre des photos d'une ligne noire sur une feuille blanche et voir le contraste entre les deux couleurs.

III- Le projet

a) Assemblage du robot

Dans un premier temps nous avons installé la Raspberry, la batterie et la caméra sur le Thymio afin d'apprendre de chacun de ces éléments. Une fois cela réalisé, nous avons travaillé sur la mise en œuvre du projet qui a pour but que le robot arrive à suivre un chemin sans dériver. Le chemin est sous forme d'une ligne noire dans un fond blanc

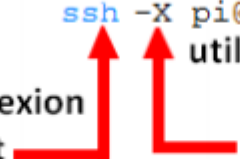
b) La connexion avec l'ordinateur

Après le branchement du robot. Nous avons démarré l'ordinateur sous linux. Ensuite, nous avons lancé la console(Shell) et entré quelques commandes pour pouvoir établir la connexion entre le robot et l'ordinateur. Nous nous sommes connectés en tapant la commande suivante :

`ssh -X pi@192.168.0.33`
utilisateur@nom_machine

commande de connexion
à un pc distant

affichage graphique à distance



Puis pour le mot de passe c'est : raspberry. Et maintenant on est sur l'interface Python, on peut accéder les fichiers qui sont enregistrés sur notre Thymio en rentrant dans le dossier Code/ Python.

Commandes utilisées :

Idle3 nom_de_fichiers : afficher le code source de fichier demandé.

Python3 nom_de_fichiers : exécuter le fichier.

asebamedulla -v "ser:device=/dev/ttyACM0" : l'assembleur (on entre & après pour continuer)

c) La connexion entre le robot et le raspberry

Cette connexion permet au Raspberry de commander les différents composants du Thymio. Pour cela a utilisé ASEBA qui est une architecture orientée événements et qui fonctionne sur des robots avec plusieurs microcontrôleurs. Plus exactement la commande ASEBAMEDULLA qui utilise la librairie Dbus pour autoriser la communication entre la Raspberry et le Thymio.

`asebamedulla -v "ser:device=/dev/ttyACM0"`

permet d'accéder à un réseau Aseba par DBus l'envoi des données par USB Définie une variable

d) Le Fonctionnement du robot

Le programme codé en python3 se compose de trois parties sous forme de classes, méthodes et objets. La première partie consiste à capturer des images à l'aide de la caméra. Bien évidemment on doit définir le temps d'enregistrement, le nombre d'images par secondes et la résolution de l'image. La deuxième partie a pour but d'initialiser la caméra et le Thymio. Et la dernière partie est celle qui commande les roues et qui analyse l'image pour déterminer leurs vitesses et le sens que le robot doit prendre pour ne pas dériver de son chemin.

Dans ce projet on a écrit une série des programmes pour que le robot suit une ligne noire (figureA) grâce à la camera (on connecte la camera avec le raspberry (figureB)) qui va prendre des photos et à chaque fois notre programme va déterminer la position de la ligne noire (le point le plus sombre).



FigureA



figureB

Comment faire rouler les roues du robot :

Pour réaliser ça nous avons utilisé le programme Thymio.py (figure7) :

```

import dbus
import dbus.mainloop.glib
from gi.repository import GObject

#
# The following command must be launch from a different console:
# aseba2dulla -v "ser:device=/dev/ttyACM0"
#
#

dbus.mainloop.glib.DBusGMainLoop(set_as_default=True)
bus = dbus.SystemBus()
bus = dbus.SessionBus()

#Create Aseba network
network = dbus.Interface(bus.get_object('ch.epfl.mobots.Aseba',
'/'), dbus_interface='ch.epfl.mobots.AsebaNetwork')

#print in the terminal the name of each Aseba NOde
print(network.GetNodesList())

#send motor value to the robot
network.SetVariable("thymio-II", "motor.left.target", [0.0])
network.SetVariable("thymio-II", "motor.right.target", [0.0])
|

```

Dans la première partie on importe les libraires nécessaires pour réaliser le programme (dbus, dbus.mainloop.glib)

Après on a créé le network aseba, finalement pour donner une vitesse aux Motors on écrit les dernières deux lignes par exemple :

```

#send motor value to the robot
network.SetVariable("thymio-II", "motor.left.target", [100.0])
network.SetVariable("thymio-II", "motor.right.target", [100.0])

```

Limites de système :

Les vitesses des roues ne peuvent pas dépasser 200.0

Comment initialiser la camera :

Pour réaliser ça nous avons utilisé le programme Camera.py (figure8) :

```

import picamera
import picamera.array

import numpy as np
import matplotlib.pyplot as plt

resolution1= 1024
resolution2=768
bound= 700
longeurMask=200

with picamera.PiCamera() as camera:
    with picamera.array.PiRGBArray(camera) as stream:
        camera.resolution = (resolution1,resolution2)
        camera.start_preview()
        time.sleep(2)
        camera.capture(stream, 'rgb')
        line=stream.array[200,0:1023,1]
        mask=np.ones([longeurMask])
        #print(line.shape)
        #plt.plot(line)
        #plt.show()
        plt.imshow(stream.array)
        plt.show()
        convolution=np.convolve(line, mask, mode='valid')
        #plt.plot(convolution)
        #plt.show()
        position=np.argmin(convolution)+(longeurMask/2)
        print(position)

```

(figure8)

Dans ce programme on a initialisé la camera puis on prend une photo en couleurs de la ligne noire, ici on veut chercher le point le plus sombre, c'est le point qui a le minimum de rgb (noire : 0rgb), en réalise ça en prenant une ligne horizontale du graph de rgb avec *stream.array*.

Programme principal de fonctionnement :

Tout d'abord on importe les library neccesaire, puis initialise les variables suivantes :

- CameraResolution : la resolution de la camera
- cameraFramerate : nombre des photo prises par seconde
- cameraRecording time : temp de notre projet
- lineIndex : la ligne horizontale ou on cherche le point le plus sombre
- longeurMask : la taille de notre mask
- positionRef : la position de reference (obtenue avec le programme de camera)
- lineBound : les limite de la ligne noire
- lineLost : la valeur de la difference ou on presume que la ligne était perdu
- leftspeed,rightspeed : les vitesses de base


```

import time
import picamera
import picamera.array #this may import some of numpy.

import dbus
import dbus.mainloop.glib

import numpy as np
import matplotlib.pyplot as plt

cameraResolution = [1024,768] # pixels
cameraFrameRate = 30          # images per seconds (NB using
mean on a 256x256 image leads to ~20 images/s)
cameraRecordingTime = 600     # seconds
lineIndex= 750
longeurMask = 200
positionRef = 532
lineBound=50
lineLost=150
LeftSpeed=70
RightSpeed=70

```

Ensuite, on initialise la camera pour prendre des photos, puis on initialise les roues et on met leurs roues à la vitesse de base (en negative pour que le Robot avance)

```

class ProcessingClass(picamera.array.PiRGBAnalysis):
    def __init__(self, camera):
        super().__init__(camera) # Be carefull super(). might
not work as is with python2
        self.initializeThymio()
        self.count = 0

    def initializeThymio(self):
        dbus.mainloop.glib.DBusGMainLoop(set_as_default=True)
        bus = dbus.SessionBus()
        busAseba = bus.get_object('ch.epfl.mobots.Aseba', '/')
        self.__network = dbus.Interface(busAseba,
dbus_interface='ch.epfl.mobots.AsebaNetwork')

        def setThymioSpeed(self, leftSpeed, rightSpeed):
            self.__network.SetVariable("thymio-II",
"motor.left.target", [-leftSpeed])
            self.__network.SetVariable("thymio-II",
"motor.right.target", [-rightSpeed])

```

Pour finir, Ici c'est notre boucle et on va rentrer le code principal :

```
def analyse(self, array):
    rightSpeed0 = 0
    leftSpeed0 = 0

    mask = np.ones([longueurMask])
    line = array[lineIndex, 1:1024, 0]
    line2 = np.convolve(line, mask, mode='valid')
    positionMin = line2.argmin()+longueurMask/2
    #print(positionMin)
    diff = positionRef-positionMin
    if -lineBound <= diff <= lineBound:
        self.setThymioSpeed(RightSpeed, LeftSpeed)
    elif lineBound < diff < lineLost:
        RightSpeed0 = RightSpeed + diff/10
        self.setThymioSpeed(RightSpeed0, LeftSpeed)
    elif -lineLost > diff > -lineBound:
        LeftSpeed0 = LeftSpeed + diff/10
        self.setThymioSpeed(RightSpeed, LeftSpeed0)
    elif diff > lineLost:
        RightSpeed0 = RightSpeed + diff/10
        LeftSpeed0 = LeftSpeed - diff/10
        self.setThymioSpeed(RightSpeed0, LeftSpeed0)
    elif (diff < -lineLost):
        RightSpeed0 = RightSpeed - 30
        LeftSpeed0 = LeftSpeed + 30
        self.setThymioSpeed(RightSpeed0, LeftSpeed0)
    # else :
    #     self.setThymioSpeed(0,0)
    #self.count = self.count + 1
    #print(self.count)
    #Do something.
```

Explication :

Tout d'abord on initialise deux variable (rightspeed0, leftspeed0) qui vont nous servir apres a incrementer les vitesses et acclerer les roues.

On cherche le point le plus sombre de nouveau a chaque fois une photo est prise (avec le même code de camera.py on calcule la difference entre le point trouver et le point de reference.

- le thymio est dans les bornes de la ligne noire : on fait un reset des vitesses (on redonne les valeurs de base)
- le Thymio vire à droite de la ligne : il doit tourner a gauche donc on accelere la roue droite
- le Thymio vire à gauche de la ligne : il doit tourner a droite donc on accelere la roue gauche

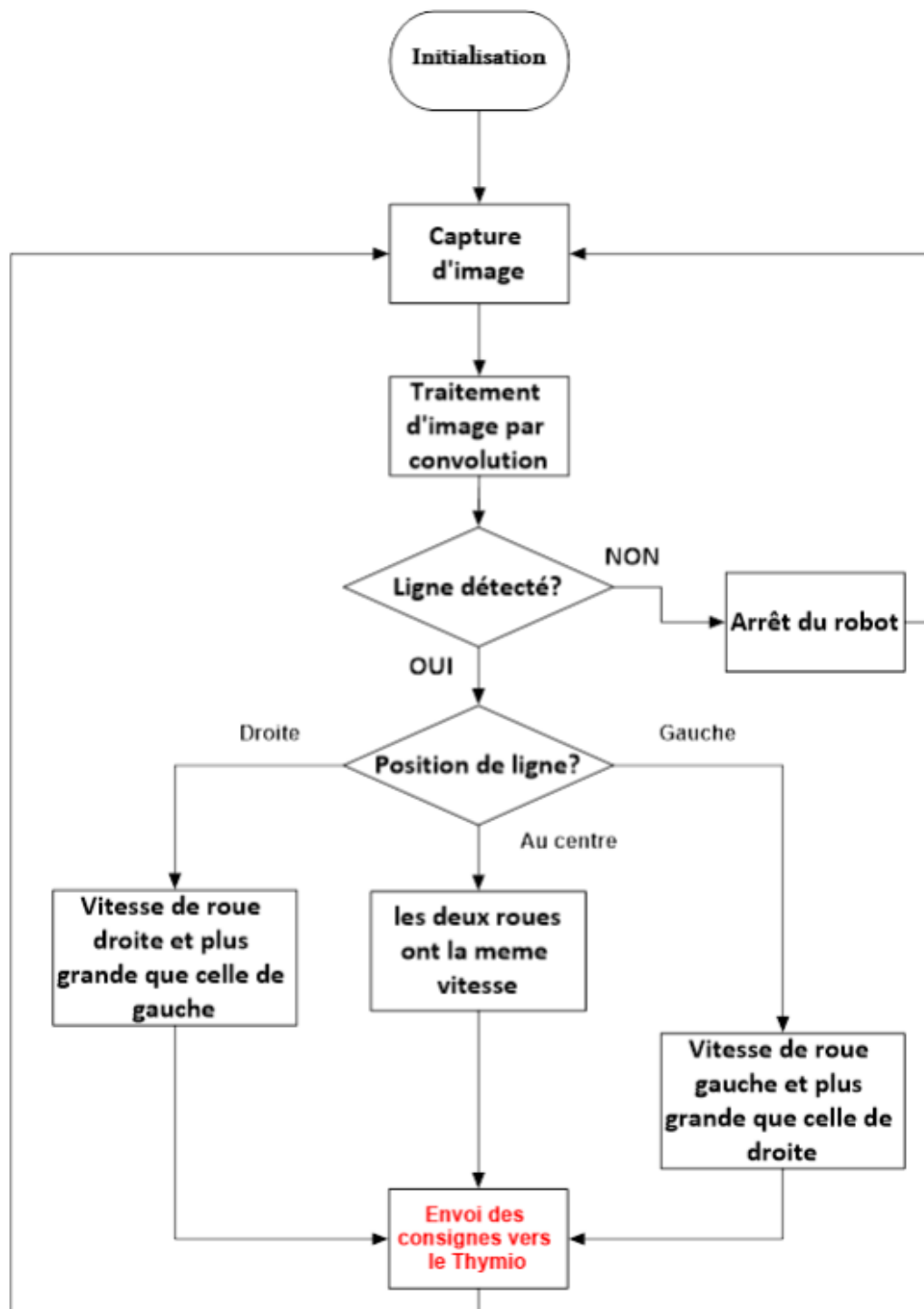
Dans cette étape on arrête la camera et on remet les vitesses à 0 :

```
# This is the part which is executed first.
#
# the " with ... as " allows to create an object
(ContextManager) which
# opens something and then be sure it is close properly
whatever happen.
# The given class (ContextManager) must have an __enter__()
and
# __exit__() method.
#
# The following snippet allows to call the analyse() method
from the class
# output every time an image is acquired.
with picamera.PiCamera() as camera:
    with picamera.array.PiRGBAnalysis(camera) as output:
        camera.resolution = (cameraResolution)
        camera.framerate = cameraFrameRate
        output = ProcessingClass(camera)
        camera.start_recording(output, format='rgb')
        camera.wait_recording(cameraRecordingTime)
        camera.stop_recording()
        output.setThymioSpeed(0, 0)
print("end")
```

IV- Conclusion

Ce projet a été très formateur cependant on aurait aimé avoir plus de 3 TP pour travailler sur le robot. Au final, Nous avons réussi à faire avancer le robot Thymio 2 et à le faire suivre la ligne grâce au programme en annexes. Nous avons mis en œuvre un programme qui permet au robot de suivre une ligne sur le sol, tout en se servant de la camera qui elle aura pour but de diriger le robot tout en détectant la ligne sur le sol, de ce fait le Raspberry pi a pour but de faire la liaison de la caméra avec le robot. C'est donc comme cela que nous avons pu mener à bien ce projet.

V- ANNEXES



Algorithme de Suivi de ligne

```

import time # on l'utilise pour les temporisations
import picamera # librairie de la caméra
import picamera.array # pour transformer l'image capturée en matrice
import dbus
import dbus.mainloop.glib
import numpy as np
import matplotlib.pyplot as plt

HeightRes = 768
WidthRes = 1024
CameraResolution = [WidthRes, HeightRes] # résolution de l'image
FramePerSecond = 15 # nombre d'images par seconde
CameraRecordingTime = 40 # temps d'enregistrement en secondes
LineIndex = 750 # la ligne qu'on convolue avec le masque
MaskLength = 200 # la longueur de masque
PositionRef = 493 # la position de référence/départ
LeftSpeed = 70
RightSpeed = 70
BigDif = 100 # The difference between the line position and PositionRef is large
SmallDif = 50 # The difference between the line position and PositionRef is small
SpeedDif = 30 # The speed difference between the wheels so that the robot can turn left or right

class Pathfinder(picamera.array.PiRGBAnalysis):
    def __init__(self, camera): # initialisation de la caméra
        super().__init__(camera)
        self.initializeThymio()

    def initializeThymio(self): # initialisation du thymio
        dbus.mainloop.glib.DBusGMainLoop(set_as_default=True)
        bus = dbus.SessionBus()
        busAseba = bus.get_object('ch.epfl.mobots.aseba', '/') # obtient le bus des moteurs du thymio
        self.__network = dbus.Interface(busAseba, dbus_interface='ch.epfl.mobots.asebaNetwork')

    def setThymioSpeed(self, LeftSpeed, RightSpeed): # méthode qui affecte les vitesses trouvées aux moteurs
        self.__network.SetVariable("thymio-II", "motor.left.target", [-LeftSpeed])
        self.__network.SetVariable("thymio-II", "motor.right.target", [-RightSpeed])

    def analyse(self, array):
        RightSpeed0 = 0
        LeftSpeed0 = 0
        mask = np.ones([MaskLength]) # remplit la matrice du masque avec des 1
        line = array[LineIndex, 1:WidthRes, 0] # prend la ligne de référence et le met dans une matrice
        line2 = np.convolve(line, mask, mode='valid') # convolution de la ligne avec le masque
        # le mode valid permet avoir le résultat de la convolution qu'on toutes les cases de la matrice du masque
        # sont convolué avec celles de la ligne
        positionMin = line2.argmin()+MaskLength/2 # position du point le plus sombre
        dif = PositionRef-positionMin # cette différence nous permet de savoir si le robot est dans le bon chemin
        # ou s'il doit corriger sa position
        if -SmallDif < =dif <= SmallDif:
            self.setThymioSpeed(RightSpeed, LeftSpeed)
        elif SmallDif < dif < BigDif:
            RightSpeed0 = RightSpeed + SpeedDif
            self.setThymioSpeed(RightSpeed0, LeftSpeed)
        elif -BigDif < dif < -SmallDif:
            LeftSpeed0 = LeftSpeed + SpeedDif
            self.setThymioSpeed(RightSpeed, LeftSpeed0)
        elif dif > BigDif:
            RightSpeed0 = RightSpeed + SpeedDif
            LeftSpeed0 = LeftSpeed - SpeedDif
            self.setThymioSpeed(RightSpeed0, LeftSpeed0)
        elif dif < -BigDif:
            RightSpeed0 = RightSpeed - SpeedDif
            LeftSpeed0 = LeftSpeed + SpeedDif
            self.setThymioSpeed(RightSpeed0, LeftSpeed0)

with picamera.PiCamera() as camera: # c'est la partie qui s'exécute en premier
    with picamera.array.PiRGBAnalysis(camera) as output:
        camera.resolution = (CameraResolution) # résolution de l'image
        camera.framerate = FramePerSecond # image capturée par secondes
        output = Pathfinder(camera) # appelle l'objet "camera" qui permet d'initialiser la caméra
        camera.start_recording(output, format='rgb') # lance l'enregistrement
        camera.wait_recording(CameraRecordingTime)
        camera.stop_recording()
        output.setThymioSpeed(0, 0)
print("end")

```

Programme principal