S. Lavirotte
Polytech Nice Sophia
SI5
2021-2022

TD n°3
Une plateforme mobile

Le but de ce TD est d'installer l'ensemble des briques logicielles pour donner vie à votre petit robot. Mais avant cela, il va falloir le construire car il vous a été fourni en kit…

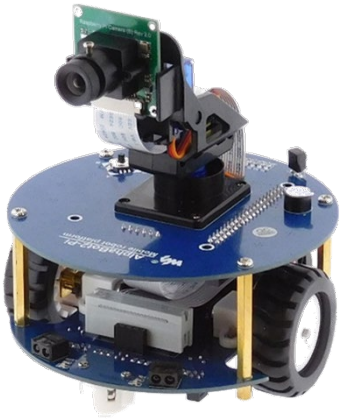
1 Construction du Robot

Comme vous êtes les premiers à pouvoir bénéficier de ce matériel, vous allez aussi avoir le plaisir (ou pas) de la construire. Pour cela, rien de plus facile en suivant les guides proposés :

- Document: https://www.waveshare.com/w/upload/1/1a/Alphabot2-pi-assembly-diagram-en.pdf
- Vidéo: https://www.youtube.com/watch?v=ONgoqpxYWQo&t=208s

Attention ! Voici quelques points de vigilance pour ne pas devoir démonter et re-monter le robot :

- Il faut bien remplacer la nappe vidéo reliée à la caméra par la nappe plus longue fournie dans le kit sous peine de constater que la nappe d'origine est trop courte pour permettre les mouvements de la caméra (oui, ça sent le vécu).
- Ne pas mettre d'entretoise à l'arrière droite entre les deux plateformes du robot afin d'avoir accès plus facilement à l'alimentation de la Raspberry Pi pendant les phases de développement où vous n'utiliserez pas le robot sur batteries.



Vous devriez en quelques minutes arriver au résultat ci-contre.

2 Installations et configurations

2.1 Installation

Afin de gagner un temps précieux, nous avons réalisé une image déjà mise à jour et incluant toutes les installations nécessaires pour le pilotage du robot. Pour en bénéficier, il vous suffit de télécharger l'image suivante et de la flasher sur la carte SD fournie avec le robot (cf TD n°1).

`http://trolen.polytech.unice.fr/cours/sia/td03/2021-10-30-raspios-bullseye-armhf-lite-ab2.7z`

Vous n'oublierez pas d'ajouter les fichiers `wpa-supplicant.conf` et `ssh` dans la partition de boot pour respectivement donner la configuration de connexion sur le réseau Wi-Fi et d'activer le serveur SSH pour vous connecter à distance.

2.2 Activation du support matériel

Une fois connecté sur la Raspberry Pi, vous commencerez par activer les différents composants matériels via la commande `sudo raspi-config`.

Dans « *Interface Options* », activez :

- La caméra
- Le bus SPI
- Le bus I2C
- Le port Série

Dans « *Advanced Options* », sélectionnez « *Expand Filesystem* » pour occuper toute la place disponible sur la carte SD.

Puis redémarrez votre Raspberry Pi. Votre robot est maintenant prête à l'utilisation !

TD n°3
Une plateforme mobile

3 Infrastructure logicielle du Robot

L'infrastructure du robot est réalisée grâce à des micro-services pour l'accès à chacune des fonctionnalités de base. Ces micro-services sont déployés via des conteneurs docker. Vous trouverez ci-dessous la liste des conteneurs et leur fonctionnalités. Nous considérons dans les exemples de code que l'adresse IP du robot est : `192.168.27.1`.

Nous allons commencer par créer un réseau pour relier nos conteneurs docker :

```
docker network create ab2-network
```

3.1 Web-control : une première image pour tester l'ensemble des fonctionnalités

Avant de commencer les développements, un premier service de test peut être mise en œuvre afin de piloter le robot manuellement, via une interface Web, tout en ayant accès au flux vidéo de la caméra. Cela vous permettra ainsi de faire un premier test pour savoir si votre robot est bien fonctionnel.

Vous devez donc récupérer/mettre à jour l'image suivante :

```
docker pull ubiquarium/ab2-webcontrol
```

Le code intégré dans cette image se trouve à l'adresse suivante :

```
git clone https://ubinas.polytech.unice.fr:38443/platform/alphabot2-pi/ab2-webcontrol.git
```

Consultez le fichier `docker-compose.yml` qui permet de démarrer cette image via la commande :

```
docker-compose up -d
```

Vous constaterez que le container `web-control` utilise un autre conteneur `ab2-mjpg-streamer` qui démarre un serveur pour l'accès au flux vidéo de la caméra. Ce flux vidéo est disponible via le port `8080`. Vous pouvez accéder à ce flux via différents moyens :

- Navigateur : `http://192.168.27.1:8080/`
- VLC : `http://192.168.0.30:8080/?action=stream`

Le conteneur `web-control`, quant à lui, expose une interface web à l'adresse suivante :

```
http://192.168.27.1:8000/
```

Une fois ces premiers tests réalisés, arrêtez le micro-service à l'aide de la commande :

```
docker-compose down
```

3.2 Core : le cœur des fonctionnalités du robot

Un micro-service permet d'accéder aux fonctionnalités principales du robot à savoir les actionneurs que sont les moteurs, servos moteurs pour le control de la caméra, le buzzer, mais aussi les capteurs que sont le joystick, les infrarouges pour l'évitement d'obstacle ou les infrarouges pour le suivi de ligne.

L'image à récupérer/mettre à jour est la suivante :

```
docker pull ubiquarium/ab2-core
```

Le code permettant cette image se trouve à l'adresse suivante :

```
git clone https://ubinas.polytech.unice.fr:38443/platform/alphabot2-pi/ab2-core.git
```

A l'aide de la commande `docker-compose up -d`, un broker MQTT sera lancé ainsi qu'un conteneur vous permettant d'envoyer ou de recevoir les informations via le broker. Vous consulterez la documentation dans le projet `ab2-core` pour connaître les topics MQTT et les valeurs véhiculées sur ces différents topics :

```
https://ubinas.polytech.unice.fr:38443/platform/alphabot2-pi/ab2-core
```

TD n°3
Une plateforme mobile

3.3 Leds RGB : une autre fonctionnalité un autre matériel

Les concepteurs du robot ont trouvé intéressant d'intégrer à la plateforme des leds RGB. Nous avons réalisé une image permettant de tester celles-ci.

L'image à récupérer/mettre à jour est la suivante :

```
docker pull ubiquarium/ab2-rgbleds
```

Le code permettant de construire cette image se trouve à l'adresse suivante :

```
git clone https://ubinas.polytech.unice.fr:38443/platform/alphabot2-pi/ab2-rgbleds.git
```

En lançant cette image à l'aide de la commande suivante, l'application de test fournie, allumera les leds RGB :

```
docker run --rm --privileged ubiquarium/ab2-rgbleds:latest lowlevel.py
```

Pour piloter les leds RGB via un broker MQTT, il suffira de lancer la commande `docker-compose up -d`. Mais attention, il devra y avoir un broker accessible localement (qui sera démarré automatiquement avec l'étape suivante).

Vous consulterez la documentation suivante pour contrôler les leds RGB via MQTT :

https://ubinas.polytech.unice.fr:38443/platform/alphabot2-pi/ab2-rgbleds#mqtt-topics

4 Validation de l'infrastructure de micro-services

4.1 Mettre en œuvre un petit automatisme pour se faire la main

Pour le moment, les différents micro-services ne font que mettre à disposition l'accès au matériel. Il est temps de mettre en place un petit programme mettant en œuvre le matériel. Nous vous proposons de faire un programme simple qui allume deux des quatre leds RGB en rouge (équivalent de feux stop) lorsque l'on envoie la commande `stop` au robot sur le topic `alphabot2/actuators/move`.

Pour faciliter la mise en œuvre de ce premier programme, vous consulterez le code suivant permettant à un programme d'envoyer des commandes ou de lire les valeurs de capteurs sur les différents topic MQTT :

https://ubinas.polytech.unice.fr:38443/platform/alphabot2-pi/ab2-core/-/blob/master/app/client.py

Pour observer ou envoyer des message sur le broker MQTT du robot, vous pouvez utiliser le programme MQTT Explorer disponible sous tous les environnement :

http://mqtt-explorer.com/

4.2 Accès concurrent au matériel

Démarrez les deux micro-services `ab2-core` et `ab2-webcontrol`. Envoyez une commande via MQTT pour faire avancer le robot. Contrôlez le robot manuellement à partir de l'interface Web. Jetez un coup d'œil sur le code fonctionnant dans ces deux conteneurs.

Voici quelques questions de réflexions au sujet de ces deux micro-services

- Est-ce que les deux micro-services permettant de piloter les roues du robot fonctionnent bien malgré la concurrence?
- Pouvez-vous justifiez ce fonctionnement ?
- Est-ce qu'un accès concurrent au matériel est toujours possible quel que soit le bus matériel ? Citez au moins un exemple pour lequel cela peut fonctionner et un autre où ce n'est pas le cas.
- Cela vous semble-t-il une bonne architecture (avoir un accès concurrent direct au matériel) ?
- Que pourriez-vous proposer comme architecture pour améliorer la situation ?

Vous verrez la semaine prochaine la mise en œuvre plus propre de cet accès pour piloter le robot.