SAE32: rapports

Bachelier Baptiste
baptiste.bachelier@etu.umontpellier.fr

A conversational robot is a specialized agent capable of conducting a conversation. The terms conversational agent or dialogist are also used. It is also defined as a human-machine interface specialized in natural language dialogue with a human user.

I. Découverte du sujet

La SAE32 commence par la découverte du sujet, a savoir la mise en service de robot conversationnel. Sur les trois robot que l'IUT a acheté il y en a un qui est a était monté par M.Bory, ainsi il ne nous en reste deux a monter.

Dans un premier temps on c'est répartie les taches à effectuer, la constructions d'autre robot la programmations des différentes composante, le déplacement, la détection

II. Mise en service d'un des robot

Afin de construire un nouveau robot, avec Aurélien et Mehdi on a fait l'inventaire des pièces que l'on a mis dans un carton vide, ainsi on peut trouver les pièces assez facilement.

Ensuite, on a commencé la construction du robot, ça a étais compliqué car tous les trous étais au mauvais endroit avec de mauvaise dimension. C'est la que les remarque qu'Alexis Bory a écrit dans sa doc de construction des UBBO ont pris sens. Le même jours on a appris que l'entreprise a l'origine de ses robot ont fait faillite et donc que l'application permettant de contrôler le robot ne marche plus.

Pour tenter de réglé se problème, j'ai tenté de comprendre le code Arduino donnée ainsi que les moyen de communication entre les cartes. Pour le code, j'ai appris que la tablette devait envoyer des instructions à l'Arduino via la carte Bluetooth connecté a l'Arduino méga.

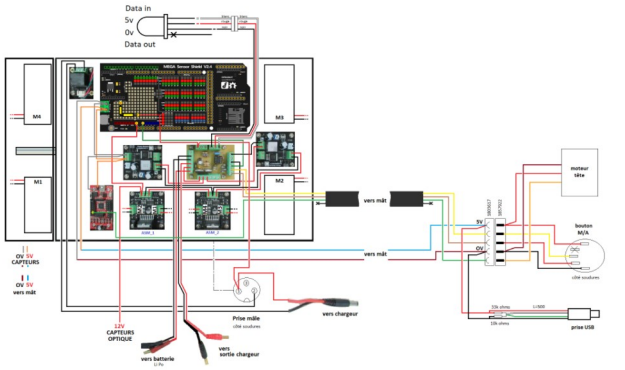


schéma electrique du robot UBBO

Le code avait la forme d'une suite de condition qui sont activé selon le code que l'on envoie, mais ça n'utilisai pas de if mais match/case.

```
const int GO_FORWARD = 0x01;
const int GO_BACKWARD = 0x02;
const int STOP = 0x03;
const int GO_BACK_DOCKSTATION = 0x4;
const int TRANSLATE_LEFT = 0x08;
const int TRANSLATE_RIGHT = 0x09;
const int MOVE_TABLET = 0x10;
const int HEART_BEAT = 0x11;
const int TURN_LEFT = 0x30;
const int TURN_RIGHT = 0x31;
```

Screen des codes d'actions dans le code initial

Le code a était compliqué à comprendre, d'une part parce que je ne connaît pas le langage et d'une autre pars car il y avait pas de commentaire, il m'as fallut plusieurs jours pour le comprendre en majorité, de même pour le schéma électrique ou le rôle de certaine carte ne m'était pas claire.

Après ça, comme le code de Rayan fonctionnait j'ai proposez quelque changement dans son afin de « copier » certaine partie du code de base. Ainsi on avait un robot qui pouvait normalement bouger sur commande, mais il y avait un problème, les codes d'actions que l'on avait mis ne marchait pas. En affichant les valeurs dans le terminal de l'IDE d'Arduino on c'est rendu compte que le code était transmis dans son code ASCII ou UNICODE, donc quand on envoyé 1 l'Arduino recevait 49.

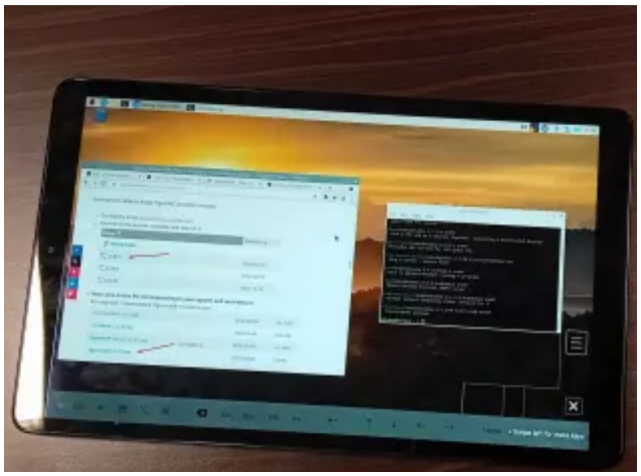
Je me suis aussi rendu compte que l'une des cartes dans le robot ne servait a rien a part complexifier le code. La carte en question est celle qui s'occupe de piloter le servomoteur, il n'y avait que le pin PWM de brancher dessus. J'ai donc débrancher le pin PWM et je les mis sur l'Arduino méga qui a des ports fait exprès pour ça, donc on a gagner 3 carte pour brancher des servomoteurs.

Après avoir modifier le code pour que ça fonctionne je me suis occuper de steamer le bureau d'une Raspberry sur une tablette. Ainsi on peut coder des actions ou des animations en python tel que parler a voix haute (grâce à une enceinte connecté en Bluetooth). Pour faire cette ça j'ai brancher la Raspberry sur un écran, avec un clavier azerty et une souris, puis je suis aller dans la config pour transformer la Raspberry en serveur VNC.

Ensuite j'ai installer un client VNC sur la tablette, directement le .apk car on ne pouvais pas accéder a GooglePlay, le mot de passe associer à l'adresse mail que l'IUT a crée pour eux. Je pense que l'on pouvait plus se connecter au compte gmail car il a était crée il y a plus de

6ans et a mon souvenir Google a fait en sorte que les compte gmail inutilisé depuis plus de 2ans soient désactivé/supprimé.

Ensuite, j'ai transformé la Raspberry en point d'accès wifi pour que le client et le serveur VNC soient sur le même réseau. Ainsi le réseau de l'IUT ne déconnecterait pas la tablette toute les 10min et personne ne pourrait venir se connecter en ssh sur la Raspberry pour modifier des choses.



Screen du client connecter au server VNC

PARTIE 2 :

Afin de continuer la mise en service des robot UBBO on a eu 4 jour de plus. L'objectif est de les mettre en état de marche pour la JPO du 8 février.

Premièrement, on a décider de faire un Kanban, pour s'avoir les taches que l'on aller faire, ainsi que la répartition des tâches.

Ainsi dans un premier temps, j'ai crée un dépôt git pour que l'on puisse centraliser et garder une version a jour de ce que l'on a fait: https://github.com/Ssea2/UBBO_iut_b-ziers

Ensuite je suis aller aidez Aurélien pour monter le second robot, comme il avait monté toute la base motorisé il ne manqué plus qu'a rajouter les tiges et le servomoteur qui contrôle l'inclinaison de la tablette, pour faire ça on a du improviser des solutions, par exemple pour emboîter un roulement a bille dans la structure on l'as mis entre 2 planche de bois (pour éviter d'abîmer la table et la structure du robot) puis on a utiliser un marteau pour l'enfoncer.
De plus, certaine vis n'était pas présente, notamment des vis américaine introuvable mais nécessaire pour visser le support de la tablette au reste du robot. Comme on ne trouvait pas de vis assez petite avec leur écrou Aurélien a repercé des troues pour pouvoir les visser.

Après, Mehdi a commencé a faire de la vision avec Teachable Machine, afin que le robot puisse faire une action suite a un signe de main. Pour faire ça il fallait donc une caméra sur le robot, j'ai donc commencé a cherché si l'on pouvait redirigé le flux vidéo des webcams de la tablette sur la Raspberry afin que l'on puisse l'utilisé avec le code python donnée par Teachable machine. Pour ce faire j'ai installer « Droidcam » sur la tablette cd qui a permis de la transformer caméra ip. Puis après avoir activer le débogage USB je les connecté à la tablette via un câble. Ensuite j'ai crée un script bash qui va exécuter ces commandes :

```
#!/bin/bash

apt update
apt install adb
apt install v4l2loopback-utils
apt install v4l-utils

adb forward tcp:4747 tcp:4747
modprobe v4l2loopback devices=1 video_nr=0 card_label="DroidCam"
```

screen des commandes permettant de crée un pont virtuel entre la caméra de la tablette et la raspberry

Ces commandes vont installer ,si ce n'est pas déjà fait, les packages nécessaire. Puis elles vont crée le pont entre la webcam et la raspberry, il ne reste plus qu'a mettre l'adresse de loopback 127.0.0.0 dans l'url de capture vidéo.

```
url = "http://127.0.0.1:4747/video"

cap = cv2.VideoCapture(url)

c=0
while True:
    ret, frame = cap.read()
    if not ret:
        print("Failed to grab frame")
        break

    if c==12:
        predic_move(frame)
        c=0
    # Convert from YUVJ420P if needed
    #frame = cv2.cvtColor(frame, cv2.COLOR_YUV2BGR_I420)

    cv2.imshow("DroidCam USB Stream", frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        cap.release()
        cv2.destroyAllWindows()
        break

    c+=1
```

screen d'une partie du code permettant de faire de la reconnaissance d'image.

Puis, j'ai fait en sorte que l'IHM de Rayan se lance au démarrage, c'est a se moment la que j'ai rencontrer le plus de problème, j'ai tester différente manière pour planifier des taches au reboot, crontab, crée un service via /etc/inid.d et /etc/system.d/system mais ça marché pas, Mr.Roy m'a dit que ce que je veux faire peut être fait avec un fichier /etc/rc.local, et ça a marché bien que des fois ça donne une erreur <[OSErreur] can't assign ip adress>, dans se cas faut se connecter au WIFI de la Raspberry puis lancer la

commande suivante </home/pi/UBBO_iut_b-ziers/start_web.sh &> afin que l'IHM s'exécute en background même si on ferme la connexion ssh.

Pour finir, je me suis occupé de raccorder la Raspberry a l'alimentation 5V du robot, le 5v sur le pin 2 et le ground sur le pin 6 de la raspberry. J'ai utilisé pinout.xyz pour savoir qu'elle pin avait qu'elle numéro.

Photo de l'emplacement de la Raspberry

La Raspberry est a cette endroit car elle est suffisamment proche de la tablette pour pouvoir la charger via le câble USB et de la base du robot pour pouvoir être alimenté en 5v.

Suite a cette fin de SAE on a un robot opérationnel (il peut bouger via l'IHM et il parle pour présenté la formation, un second de monté mais il veut pas s'allumer, et normalement le code pour utiliser le lidar sous linux.