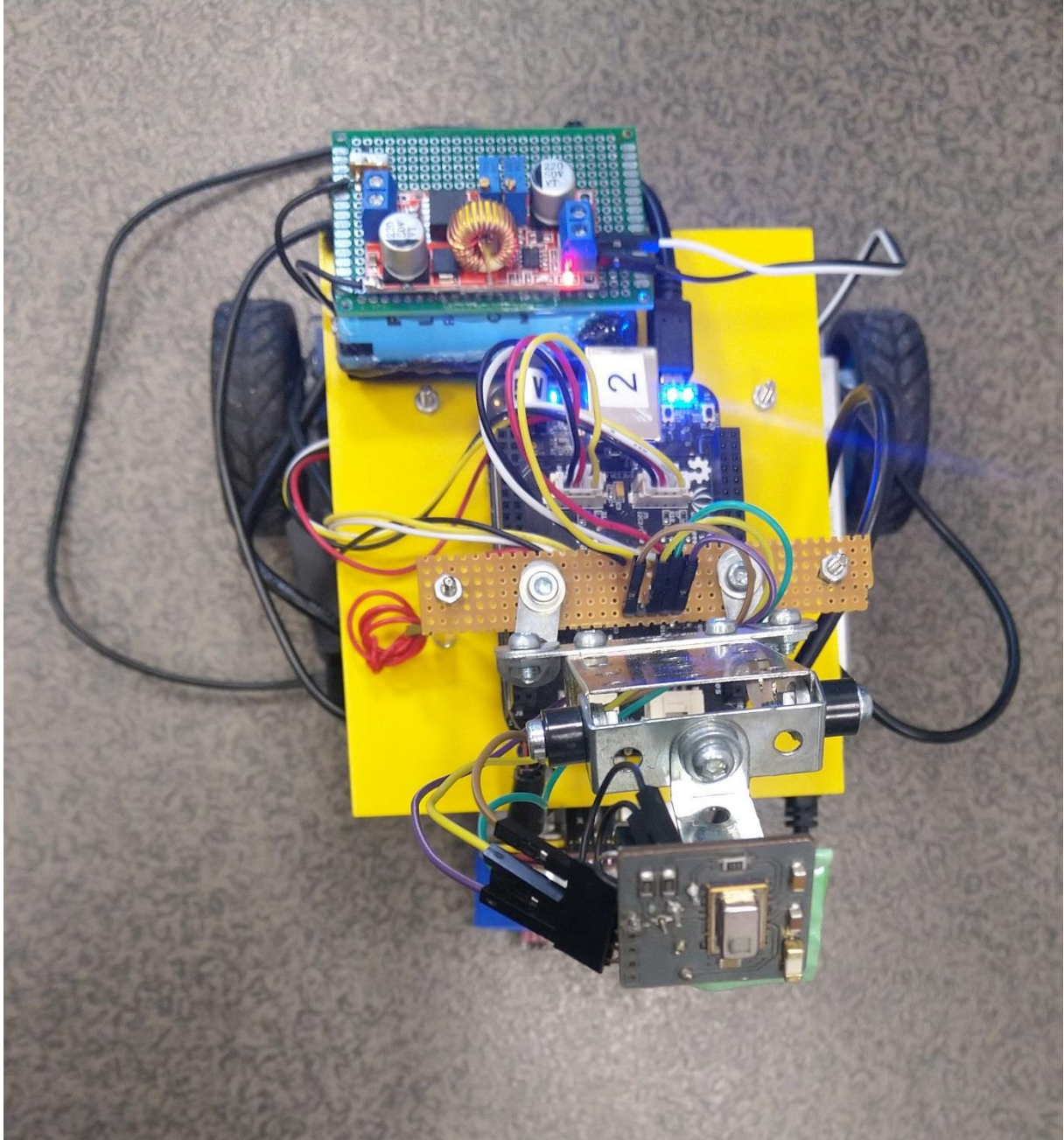


Projet : Robot à guidage thermique

Documentation technique



Julien CHAVAS

Clément DEVEVEY

Alexandre HUMBERT

Promotion FISE 2 - 2020/2021

Table des matières

I. Préambule	3
II. Configuration de la machine virtuelle	3
1) Ajouter la machine virtuelle	3
2) Automatisation de l'ajout de l'USB serial.....	3
3) Démarrage.....	3
III. Configuration et compilation du système de fichiers, noyau et device-tree	4
IV. Configuration du serveur Boa.....	5
V. Génération des exécutables	5
VI. Automatisation du lancement de service.....	5
VII. Configuration de la Carte Teensy	6
1) Prérequis logiciels.....	6
2) Commandes disponibles	6
VIII. Création d'une carte SD	8
1) Sur Linux à partir du répertoire targetfs	8
2) Avec le fichier image	8
IX. Algorithmes.....	9
1) Fonctionnement global du robot	9
2) Calibrage.....	9
X. Programmes.....	10
1) Description	10
2) moteur.c	10
3) camera.c	10
4) ultrason.c.....	10
5) robot.c	10
XI. Les programmes de débogage.....	11
1) moteur.....	11
2) camera.....	11
3) ultrason.....	11
XII. Liens utiles	12

I. Préambule

Cette documentation contient l'ensemble des étapes nécessaires pour compiler un OS Linux embarqué exécutant un programme permettant au robot thermique de suivre une personne.

L'ensemble des fichiers nécessaires est disponible dans une archive fournie avec cette documentation.

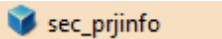
Vous pouvez retrouver cette documentation et l'ensemble de ces fichiers sur <https://github.com/alexandre-humbert/robot-thermique>

II. Configuration de la machine virtuelle

1) Ajouter la machine virtuelle

- Récupérer l'archive ZIP disponible à TSE sur Y:\mvpool\Prj_info et décompresser cette archive.

Sous Oracle VM VirtualBox :

- Machine -> Ajouter -> 
- Dans configuration -> réseau -> choisir la carte 1 NAT -> carte 2 : accès par pont (choisir votre Ethernet)
- Créer un dossier vide (mv_share) sur votre Windows
- Dans configuration -> dossiers partagés, modifier le chemin dans Dossiers permanents

2) Automatisation de l'ajout de l'USB serial

Il est possible d'automatiser le branchement du port série pour communiquer avec la carte BeagleBone :

- Alimenter la BeagleBone Black.
- Brancher l'adaptateur USB-série au PC.
- Configuration -> USB -> Ajouter un filtre USB.
- Sélectionner l'adaptateur



3) Démarrage

Utiliser le compte tpuser (mdp : tpuser) pour un utilisateur sans privilèges.

Utiliser le compte root (mdp : admintest) pour passer super-utilisateur.

- En **root** : **apt-get install net-tools**

Ceci vous permet d'utiliser ifconfig.

Rendez-vous en **root** dans `/etc/network` puis faire

- **mousepad interfaces** et recopier le contenu du fichier `interfaces` (fourni dans le zip : `fs-VM/etc/network/interfaces`)

- **reboot** afin de redémarrer la machine virtuelle avec les bonnes configurations IP.

Ceci vous permet d'avoir internet sur la machine virtuelle ainsi que d'attribuer une IP au PC (192.168.101.36) pour communiquer avec la carte BeagleBone.

- Installer et configurer un serveur NFS.

Dans cette documentation, le répertoire partagé en NFS s'appelle `/targetfs`.

III. Configuration et compilation du système de fichiers, noyau et device-tree

En **root** :

- Sous `/opt/packages/gcc` : **bash gcc_inst**
- Sous `/opt/packages/build` : **bash build_inst**

En **tpuser** :

- Sous `/sec/download` : **tar xvf buildroot.tar.gz -C /sec**
- Sous `/sec/buildroot` : **make beaglebone_defconfig**
- Sous `/sec/buildroot` : **mkdir dl**
- N'importe où : **cp -a /sec/download/dltool/* /sec/buildroot/dl**
- Sous `/sec/buildroot` : **make toolchain**
- **Copier** le fichier « `am335x-projet.dtsi` » dans :
 - o `/sec/buildroot/output/build/linux-headers-tse/arch/arm/boot/dts`
- Sous `/sec/buildroot/output/build/linux-headers-tse/arch/arm/boot/dts`
 - o **Rajouter** la ligne : **#include "am335x-projet.dtsi"** dans le fichier « `am335x-projet.dts` ».
- Sous `/sec/buildroot/output/build/linux-headers-tse/arch/arm/boot/dts` exécuter :
 - o **make**
- Sous `/sec/buildroot/` exécuter :
 - o **make menuconfig**
 - **Cocher** Target packages -> Networking application -> [*]**Boa**
 - **Cocher** Filesystem -> Filesystem images -> [*] **tar the root filesystem**
 - o **make device-tree**
 - o **make kernel**
 - o **make filesystem**

En **root** :

- Sous `/sec/buildroot/output/images` exécuter :
 - o **tar xvf rootfs.tar -C /targetfs**
 - o **cp zImages /targetfs/boot**
 - o **cp am335x-boneblack.dtb /targetfs/boot**

Actions effectuées sur le device-tree :

- Activation de l'UART1 :
 - o Communication avec la carte teensy
- Activation I2C1 et I2C2
 - o Communication avec le capteur thermique
 - o Communication avec le capteur ultrason

IV. Configuration du serveur Boa

En **root** :

Copier le contenu du dossier **mini-site** dans /targetfs/var/www/

Copier le fichier de configuration boa.conf du répertoire fs-BB dans /targetfs/etc/boa/boa.conf

Copier le script fourni « fs-BB/usr/lib/chi-bin/interface.cgi » dans /targetfs/usr/lib/cgi-bin (créer le répertoire s'il n'existe pas)

Donner les droits d'exécution : `chmod 777 /targetfs/usr/lib/cgi-bin/interface.cgi`

V. Génération des exécutable

Récupérer tous les fichiers sources disponibles dans le dossier programme.

Pour que le make s'effectue, il faut créer un dossier build (**mkdir build** en **tpuser**) dans le répertoire programme. Ce dossier contiendra tous les exécutable et fichiers objets (.o) générés par le Makefile.

Il ne vous reste plus qu'à make !

Ajouter /usr/local/bin au PATH. Vous pouvez directement copier le fichier fs-BB/etc/profile dans /targetfs/etc/

VI. Automatisation du lancement de service

Récupérer les fichiers disponibles dans /fs-BB/etc/init.d/ et les copier dans /targetfs/etc/init.d

Mettre des droits d'exécution à ces fichiers : (en root) `chmod 777 S03robot rcS S02boa`

Ces opérations vous permettent de lancer des services qui exécutent vos programmes au démarrage de la carte BeagleBone Black. **Attention à ne pas laisser votre robot sur la table lors du démarrage.**

VII. Configuration de la Carte Teensy

1) Prérequis logiciels

Nous avons créé notre propre fichier .ino à la suite de problèmes d'asservissement des moteurs. Il n'y a plus d'asservissement mais des coefficients pour limiter les différences de vitesse entre les deux roues. Notez que ce correctif ne fonctionne pas en marche arrière.

Pour compiler et téléverser ce fichier sur la carte Teensy, il faut utiliser les logiciels [Teensyduino](#) avec [Arduino](#) :

- 1) Ouvre le fichier .ino avec ArduinoIDE
- 2) Branchez la Teensy au PC en USB
- 3) Dans ArduinoIDE : Outil -> Type de carte -> Choisir Teensy 3.2/3.1
- 4) Choisir son port dans Outils -> Port
- 5) Cliquer sur la flèche pour téléverser (Teensyduino s'ouvre si la compilation réussit)

Ce fichier inclut d'autres fichiers .h et .c, attention à bien les mettre dans le dossier « arduino/library ». **Sinon** changer les #include<fichier.h> en #include"fichier/fichier.h" et les mettre dans le même dossier que le fichier .ino.

2) Commandes disponibles

Via l'**UART1** (/dev/ttyO1) on va pouvoir envoyer des commandes (sous forme de deux caractères) à la Teensy depuis la BeagleBone. [Exemple](#) : echo av > /dev/ttyO1

Voici le tableau récapitulatif des commandes disponibles :

Nom commande	Action
av	Avancer
st	Stop
ar	Reculer
dr	Avancer droite
ga	Avancer gauche
ad	Reculer droite
ag	Reculer gauche
td	Tourner gauche
tg	Tourner gauche
pd	Pivoter droite
pg	Pivoter gauche

Pour les commandes av et ar uniquement :

- "av n" permet d'avancer à la vitesse n (n compris entre 0 et 255, de préférence entre 120 et 255).
 - "av n m" permet d'avancer à la vitesse n pour la roue droite et m pour la roue gauche.
 - "av 1", "av 2", "av 3" et "av 4" permettent d'avancer à des vitesses prédéfinies.
-
- "ar n" permet de reculer à la vitesse n (n compris entre 0 et 255, de préférence entre 120 et 255).
 - "ar n m" permet de reculer à la vitesse n pour la roue droite et m pour la roue gauche.
 - "ar 1", "ar 2", "ar 3" et "ar 4" permettent de reculer à des vitesses prédéfinies.

VIII. Création d'une carte SD

1) Sur Linux à partir du répertoire targetfs

- Pourquoi créer une carte SD : une fois le programme au point, vous pourrez recréer un linux embarqué afin que votre programme puisse tourner sur le robot directement au démarrage de la carte. Voici comment procéder :
- Trouvez quel périphérique correspond à votre carte SD avec la commande : **ls /dev**
- Sur la VM, ce périphérique sera sûrement **/dev/sdd**
Les instructions suivantes sont données pour un disque **sdd**. Adaptez les commandes si votre chemin est différent.
- Ensuite **en root**:
 - Effacez la table de partition : **dd if=/dev/zero of=/dev/sdd bs=1M count=1**
 - Créez une nouvelle table : **sfdisk -H 255 -S 63 /dev/sdd << EOF**
0,1,c,*
,2
EOF
 - Ensuite créez la partition de boot : **mkfs.vfat /dev/sdd1 -n boot**
 - Créez le point de montage : **mkdir /media/boot**
 - Puis Montez cette partition : **mount /dev/sdd1/boot /media/boot**
 - Copiez les fichiers sur cette partition : **cd /sec/buildroot/output/images**
cp MLO zImage u-boot.img am335x-boneblack.dtb /media/boot
 - Créez la seconde partition : **mkfs.ext2 /dev/sdd2 -n**
 - Montez cette partition puis placez-vous dans le répertoire
 - Copiez votre système de fichier : **cp -r /targetfs/* ./**

Enfin, il faudra aller du côté de /etc/init.d et créer un service qui lancera votre programme au démarrage. (Voir dans fs-BB -> etc -> init.d pour plus d'informations)

2) Avec le fichier image

Sur Linux :

Télécharger **sdcard.img** et installer avec la commande : **dd bs=1M if=sdcard.img of=/dev/sdd**

Sur Windows :

Il est possible d'installer l'image avec un logiciel comme [Win32Disk](#).

Télécharger **sdcard.img** (disponible dans le dossier source) puis choisissez ce fichier dans le logiciel, sélectionnez votre carte SD et cliquez sur "Write".

IX. Algorithmes

1) Fonctionnement global du robot

Pour le suivi thermique, on utilise un capteur thermique possédant une matrice de 8x8 pixels. Sachant que l'on veut suivre une personne, donc un « objet » vertical, on se fît aux colonnes de la matrice en faisant la moyenne de chaque colonne. Ensuite si la moyenne la plus élevée dépasse le seuil de déclenchement, elle va donc être traduite en une instruction aux moteurs suivant la colonne. Sinon c'est du bruit et on attend.

Si le capteur ultrason détecte un obstacle à moins de 15cm alors :

- Si la matrice détecte une source de chaleur de proche, c'est-à-dire que moyenne total de la matrice du capteur thermique dépasse un seuil : $8 \times 5 \times 27.25 + 8 \times 3 \times$ la température ambiante.

Ce seuil est à régler en affichant `camera.sum_pix` car il diffère pour chaque environnement.

- Si la matrice ne détecte pas une source de chaleur proche, le robot effectue une séquence prédéfinie pour éviter l'obstacle, pendant cette séquence le robot reste capable de s'arrêter en cas d'objet proche.

Si le robot ne détecte pas de source de chaleur, il pivote sur lui-même.

La vitesse s'adapte selon la distance à laquelle se trouve l'obstacle ou l'humain. Plus le robot est proche, plus il ralentit.

Si le robot est connecté en Ethernet, un site web Boa peut servir à visualiser les informations issues des capteurs.

- Des flèches permettent le fonctionnement du robot manuellement
- Un affichage de la matrice avec la température
- Affichage température ambiante.
- Un affichage couleur en fonction de la température minimum
- Le retour de distance du capteur ultra-son
- Possibilité de modifier :
 - o Le min et le max pour régler les couleurs
 - o Taper une commande au clavier (av, ar, [cf le tableau ici](#)).
 - o Afficher ou non les températures sur chaque carré de la matrice

2) Calibrage

Le programme est calibré pour 25° de température ambiante (afficher par le capteur). Pour une température inférieure, le fonctionnement sera le même. En revanche si la température est plus élevée, le programme sera moins performant car la différence de température avec le corps sera moindre. De plus, il faudra ajuster le seuil « COEF_TEMP » dans « camera.h ». Par exemple pour une température ambiante de 30° affichée par le capteur, il faut un « COEF_TEMP » de 1,2.

Par défaut le « COEF_TEMP » est réglé à 1,04.

X. Programmes

1) Description

Le programme en langage C permet au robot de suivre une personne avec une matrice thermique. Il est composé de 4 fichiers .c avec chacun leur fichier .h, pour un total de 8 fichiers.

- Le fichier moteur
- Le fichier camera
- Le fichier ultrason
- Le fichier robot

2) moteur.c

Il contient tout le code permettant de communiquer avec la Teensy en UART. Il permet d'avancer, reculer, s'arrêter ainsi que de suivre des parcours.

3) camera.c

Il contient le code pour acquérir une image depuis la matrice thermique en I2C et le code pour exploiter ces données.

4) ultrason.c

Il contient le code pour mesurer une distance avec le capteur à ultrason.

5) robot.c

Il contient le programme principal. Il permet de piloter le robot en fonction des données des capteurs.

XI. Les programmes de débogage

1) moteur

Ce programme permet d'envoyer des commandes aux moteurs.

Exemples :

- "moteur av" permet d'envoyer la commande avancer.
 - "moteur av 1" permet d'envoyer la commande avancer à la vitesse 1.
- Se référer à la partie "[Configuration de la carte Teensy](#)" de la documentation pour plus d'information sur les commandes.

2) camera

Ce programme permet d'afficher une image de la caméra sous forme de matrice ainsi que la température ambiante et la luminosité de la pièce.

Il dispose de plusieurs options :

- -h pour afficher l'aide
- -t pour ajouter un délai
- -m pour choisir une température minimale
- -M pour choisir une température maximale
- -w pour afficher une image au format SVG

3) ultrason

Ce programme permet d'afficher la distance et la luminosité mesurées par le capteur ultrason. Il dispose de plusieurs options :

- -h pour afficher l'aide
- -t pour ajouter un délai
- -g pour changer le gain
- -r pour changer la portée
- -w pour renvoyer un texte formaté pour le web

XII. Liens utiles

- [Grove Cape for BeagleBone](#)
- [Grid-Eye Panasonic](#)
- [BeagleBone Black](#)
 - [Schéma des GPIO](#)
- [Capteurs à ultrasons SRF08](#)