Table des matières

[Configuration de la Carte BeagleBoneBlack 2](#_Toc62062137)

[1°) Inclusion des fichiers 2](#_Toc62062138)

[2°) Reconstruction 2](#_Toc62062139)

[3°) Configuration 2](#_Toc62062140)

[Action effectué 2](#_Toc62062141)

[Installation du serveur Web Debug (Boa) 2](#_Toc62062142)

[1°) Utilisation de l’archive Boa-tse : 3](#_Toc62062143)

[2°) Configuration dans le répertoire cible : 3](#_Toc62062144)

[Configuration de la Carte Teensy 3](#_Toc62062145)

[Algorithme 3](#_Toc62062146)

[1°) Fonctionnement global du robot 3](#_Toc62062147)

[2°) Calibrage 4](#_Toc62062148)

## Configuration de la Carte BeagleBoneBlack

### 1°) Inclusion des fichiers

* Copier le fichier « am335x-projet.dtsi » dans :
  + /sec/buildroot/output/build/linux-tse/arch/arm/boot/dts
* Ajouter la ligne dans « am335x-boneblack.dts » :
  + #include «am335x-projet.dts

### 2°) Reconstruction

* Dans /sec/buildroot/output/build/linux-tse/arch/arm/boot/dts executer:
  + make
* Dans /sec/buildroot/output/images executer:
  + tar xvf rootfs.tar -C /targetfs
  + cp zImages /targetfs/boot
  + cp am335x-boneblack.dtb /targetfs/boot

### 3°) Configuration

* Ajouter la ligne dans « ect/exports » :
  + /targetfs \*(rw, no\_root\_squash, sync, no\_subtree\_check)
* Executer:
  + nfs-kernel-server

### Actions effectuées

* Activation de l’UART1 :
  + Communication avec la carte teensy
* Activation I2C1 et I2C2
  + Communication capteur thermique
  + Communication capteur ultrason

## Installation du serveur Web Debug (Boa)

### 1°) Utilisation de l’archive Boa-tse :

* Décompression package avec tar xvf

### 2°) Configuration dans le répertoire cible :

* Lancer : make
* ./configure CC= « chemin du cross compilateur » CFLAGS= « -Os -pipe -Wall »--host=arm
* Lancer : make
* On copie l’exécutable Boa dans un dossier partagé en nfs avec la carte.
* On monte le le dossier de partage :
  + mount -t nfs -o nolock @IPServeur:/ « dossier partagé » /usr/local/bin
* On copie le boa.conf fourni le dossier partagé
* On créer un répertoire dans le dossier partagé puis on copie le cgi fourni dedans
  + mkdir lib/cgi-bin/

## Configuration de la Carte Teensy

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 supprimer les différences de vitesse entre les deux roues.

Pour compiler et téléverser ce fichier sur la carte Teensy, il faut utiliser les logiciels [**Teensyduino**](https://www.pjrc.com/teensy/td_download.html) avec [**Arduino**](https://www.arduino.cc/en/software) :

1. Ouvre le .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 réussi)

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.h" et les mettre dans le même dossier que le .ino.

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", "av 2", "av 3" et "av 4" permettent de reculer à des vitesses prédéfinies.

## Algorithme

### 1°) Fonctionnement global du robot

Le robot, pour effectuer son suivit thermique, utilise un capteur thermique utilisant 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 colonne par colonne. Ensuite si la moyenne la plus élevé dépasse le seuil de déclanchement, elle va donc être traduit en une instruction au moteur.

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

* Si la matrice détecte une source de chaleur de proche, avec la matrice du capteur est « pleine » c’est-à-dire que moyenne total de la matrice du capteur thermique dépasse un seuil :   
  8 x 8 x température ambiante x COEF\_TEMP
* 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 est connecté en Ethernet un site web Boa peut servir de débug.

* Des flèches permettent le fonctionnement du robot manuellement
* Un affichage de la matrice avec la température
* Un affichage couleur en fonction de la température minimum
* Le retour de distance du capteur a ultra-son

### 2°) Calibrage

Le programme est calibré pour 25° de température ambiante (afficher par le capteur), pour une température inférieure à cette dernière, le fonctionnement sera le même. En revanche si la température est plus élevée, d’une le programme sera moins performant car la différence de température avec le corps sera moindre, mais aussi il faudra ajuster le seuil « COEF\_TEMP » dans « camera.h ». Par exemple pour une température ambiante de 30° afficher par le capteur, il faut un « COEF\_TEMP » de 1,2.

Par défaut le « COEF\_TEMP » est réglé a 1,04.