

Contents

1	Présentation Robot	2
2	Architecture du projet	2
2.1	Vivado	2
3	Liens utiles	3
4	Génération du bootloader FSBL	3
4.1	créer bootgen depuis xilinx	3
4.1.1	Génération boot.bin avec Vivado	3
4.1.2	Résultat génération boot.bin	3
4.2	créer bootgen avec le script genere.sh	3
5	flash l'image sur la carte SD	4
5.1	créer bootgen avec le script genere.sh	4
5.2	configuration des partitions	4
6	Présentation Robot	6
7	Présentation Télécommande	7

Intro

1 Présentation du robot

LE robot peut fonctionner soit en mode autonome, soit en mode semi-autonome, c'est à dire qu'il est piloté par une appli Android

2 Architecture du projet

Les axes de développement du projet sont

- Le linux embarqué J'utilise Yocto Linux pour faire tourner le logiciel du robot. Le Linux est généré et claqué sur une carte SD. Son rôle est de lancer le logiciel du robot, communiquer de manière sécurisée sur le réseau avec un protocole sécurisé comme SSH, gérer les mises à jours.
- Logiciel du robot est constitué de programmes qui s'exécutent sur la cible et qui permettent de faire tourner le robot. Ces Logiciels permettent de piloter les moteurs, traiter les informations des différents capteurs, gérer le mode autonome du robot, et communiquer avec la télécommande.
- L'électronique et la gestion des couches basses se font dans la partie FPGA du composant. Tous ce qui est gestion des périphériques se fait matériellement, comme la gestion des signaux électriques, les protocoles d'échange avec les différents capteurs et actionneurs. Le logiciel communique avec le FPGA avec un protocole allégé.
- L'intégration mécanique est pour le moment hyper rudimentaire

L'arborescence contiendra donc les répertoires suivants :

• 2.1 Vivado

: le projet vivado pour la partie FPGA

Linux

3 Liens utiles

<https://blog.mbedded.ninja/programming/embedded-linux/zynq/building-linux-for-the-zynq-zc702-eval-kit-using-yocto/gid=1pid=1>

<https://www.yoctoproject.org/docs/2.4/dev-manual/dev-manual.html>

4 Génération du bootloader FSBL

/home/nicolas/vivado/vivado/FPGA_xilinx/vivado2019_2/workspace/design_1wrapper_orona2/export/design_1wrapper_orona2.bit
/home/nicolas/vivado/vivado/FPGA_xilinx/vivado2019_2/workspace/dsfyoyo/Debug/dsfyoyo.elf
/home/nicolas/vivado/vivado/FPGA_xilinx/vivado2019_2/workspace/design_1wrapper_orona2/bitstream/design_1wrapper_orona2.bitstream

4.1 créer bootgen depuis xilinx

4.1.1 Génération boot.bin avec Vivado

https://www.xilinx.com/support/documentation/sw_manuals/xilinx2019_2/ug1283-bootgen-user-guide.pdf

```
nicolas@debianicolas: /yocto/lerobot/generation_boot more genere.bif
//arch = zynq; split = false; format = BIN
the_ROM_image :
[bootloader]u-boot-zybo-zynq7.elf
design_1wrapper.bit
nicolas@debianicolas: /yocto/lerobot/generation_boot
/home/nicolas/Xilinx/Vivado/Vivado/2020.1/bin/bootgen -image
genere.bif -o boot.bin -w
```

4.1.2 Résultat génération boot.bin

On obtient alors le résultat :

```
' ***** Xilinx Bootgen v2020.1
**** Build date : May 27 2020-20:33:36
** Copyright 1986-2020 Xilinx, Inc. All Rights Reserved.
[INFO] : Bootimage generated successfully Le .bit est le fichier généré
par vivado2019
le.elf est le fichier de boot généré par vitis, mais ici récupéré chez Xilinx
(pb d'horloge)
```

4.2 créer bootgen avec le script genere.sh

```
!/bin/sh /home/nicolas/Xilinx/Vivado/Vivado/2020.1/bin/bootgen -
arch zynq -image genere. bit -o boot.bin -w
```

5 flash l'image sur la carte SD

La distribution Linux doit être flashée sur une carte SD, afin que la carte du Robot puisse booter dessus. La carte doit être configurée en 2 partitions, celle contenant le boot, et celle contenant le root. La carte SD doit être bootable.

5.1 créer bootgen avec le script genere.sh

```
sudo dd if=/dev/zero of=/dev/mmcblk0 of=/dev/
```

5.2 configuration des partitions

```
sudo fdisk /dev/mmcblk0
  Command (m for help): n
    Partition type: p primary (0 primary, 0 extended, 4 free) e extended Select
  (default p): p
    Partition number (1-4, default 1): 1
    First sector (2048-15759359, default 2048): Using default value 2048
    Last sector, +sectors or +sizeK,M,G (2048-15759359, default 15759359):
+200M
    Command (m for help): n
    Partition type: p primary (1 primary, 0 extended, 3 free) e extended Select
  (default p): p
    Partition number (1-4, default 2): 2
    First sector (411648-15759359, default 411648): Using default value 411648
    Last sector, +sectors or +sizeK,M,G (411648-15759359,
  default 15759359): Using default value 15759359
    Command (m for help): a
    Partition number (1-4): 1
    Command (m for help): t
    Partition number (1-4): 1
    Hex code (type L to list codes): c
    Changed system type of partition 1 to c (W95 FAT32 (LBA))
    Command (m for help): t
    Partition number (1-4): 2
    Hex code (type L to list codes): 83
    Commande (m pour l'aide) : p
    Disque /dev/mmcblk0 : 7,4 GiB, 7948206080 octets, 15523840 secteurs
    Unités : secteur de 1 512 = 512 octets
    Taille de secteur (logique / physique) : 512 octets / 512 octets
    taille d'E/S (minimale / optimale) : 512 octets / 512 octets
    Type d'étiquette de disque : dos
    Identifiant de disque : 0xca4a050b
    Périphérique Amorçage Début Fin Secteurs Taille Id Type
    /dev/mmcblk0p1 * 2048 411647 409600 200M c W95 FAT32 (LBA)
    /dev/mmcblk0p2 411648 15523839 15112192 7,2G 83 Linux
    Command (m for help): w
    ***** creer les sytemes de fichiers
    sudo mkfs.vfat -F 32 -n boot /dev/mmcblk0p1
    sudo mkfs.ext4 -L root /dev/mmcblk0p2
```

```

***** rajouter des trucs linux
dans local.conf ajouter le truc
source oe-init-build-env
dégager ce qu'il y a dans la carte sd root
***** mettre le nouveau file system
sudo tar x -C /media/nicolas/root/ -f core-image-minimal-zybo-zynq7.tar.gz
sudo tar x -C /mnt/mmcblk0p2/ -f core-image-minimal-zybo-zynq7.tar.gz
copier le contenu du répertoire dans la partition boot
cp * /media/nicolas/boot/
sudo mount /dev/mmcblk0p2 /mnt/mmcblk0p2
*****compiler
/poky/meta-xilinxcd..
/poky
bitbake-layers add-layer "HOME/poky/meta - xilinx"
rajouter MACHINE ?= "zybo-zynq7" dans local.conf
bitbake-layers add-layer "HOME/yocto/poky/meta - xilinx"
*****generation du boot.bin avec le bitstream
//arch = zynq; split = false; format = BIN
theROMimage :
[bootloader]/home/nicolas/vivado/vivado/FPGA_xilinx/vivado20192/workspace/dsfyoyo/Debug/dsfyoyo
/home/nicolas/vivado/vivado/FPGA_xilinx/vivado20192/workspace/dsfyoyo/ide/bitstream/design1_w
/home/nicolas/yocto/poky/build/tmp/deploy/images/zybo-zynq7/u-boot.elf
/home/nicolas/yocto/poky/build/tmp/deploy/images/zybo-zynq7/u-boot-zybo-
zynq7.elf
cp *.bin /mnt/mmcblk0p1
cp *.img /mnt/mmcblk0p1
cp *.dtb /mnt/mmcblk0p1
cp *.elf /mnt/mmcblk0p1
cp *.u-boot /mnt/mmcblk0p1
bitbake -c menuconfig virtual/kernel
http://openpowerlink.sourceforge.net/web/openPOWERLINK/Getting

```

Logiciel

6 Présentation Robot

LE robot peut fonctionner soit en mode autonome, soit en mode semi-autonome, c'est à dire qu'il est piloté par une appli Android coucou

Partie logicielle

compilation du logiciel

le logiciel s'exécute sur la cible. Il est téléchargé par scp, puis recompilé par la commande

`sh compile_ione.sh`

mise au point avec gdb

`gdb` soie

`(gdb) set args simuPC = lance en mode simulation pc`

execution du logiciel

mode simulation

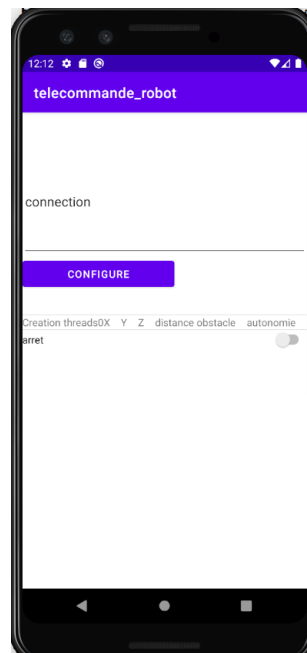
permet de simuler le fonctionnement des capteurs et ainsi tester différents scénarios. Par exemple tester un scénario avec le capteur de distance.

Pour cela taper :

7 Présentation Télécommande

LA télécommande Android permet par le réseau local de piloter le robot en mode semi-autonome.

La première page sur laquelle on arrive



Ensuite on configure le nom d'utilisateur, le mot de passe et l'adresse du robot dans la page de configuration.



Une fois les paramètres de configuration validés, on revient sur la page de commande du robot. On appuie sur connecte, et si la connection avec le robot est établie, les boutons de pilotages apparaissent.

