Contents

Intro

1 Présentation du robot

Mon projet était à la base de faire un aspirateur robot de A à Z. Bon, finalement, j'ai acheté un aspi robot tout fait. Je n'ai pas abandonné mon projet de robot pour autant, et faire un projet à ma sauce de A à Z, sans chef, sans client, sans délai à tenir, ça me plaît.
Le cahier des charges de départ est alors :

- fonctionner en mode autonome, se balader dans la maison et dans le jardin
- pouvoir être commandé avec une télécommande, pour chasser le chat du voisin un peu trop intrusif.
- être assez malin pour ne pas se manger les murs, rester coincé sous une chaise etc.

1.1 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.
- Le logiciel du robot qui 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 :

- ## 1.2 Vivado

  le projet vivado pour la partie FPGA

2 Matériel utilisé

J'utise pour le robot, un châssis récupéré d'un robot Makebloc, 2 moteurs à courant continus, un capteur de distance à ultrasons, un gyroscope et des cartes électroniques de contrôle, un modem Wifi, un coupleur de piles, des powerbanks.

2.1 Le chassis

Sur le chassis sont fixés la boite de contrôle, le modem wifi, les moteurs avec les roue, la roulette de nez.

2.2 Les moteurs

Il s'agit de 2 moteurs à courant continus placés de chaque côté, reliés directement aux roues.
Les moteurs sont pilotés par un pont en H et par modulation PWM

2.3 Le capteur de distance

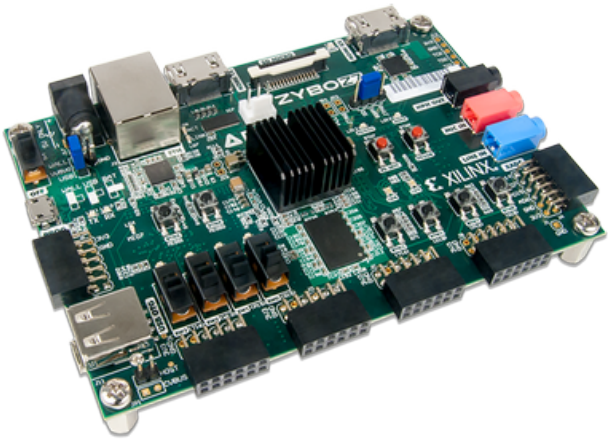
C'est la capteur ultrason qui permet de mesurer sa distance par rapport à un obstacle

2.4 Le gyroscope

Il permet de mesurer sa rotation angulaire sur les 3 axes X, Y et Z.

2.5 La carte mère

Il s'agit d'une carte Zybo de Digilent, équipée d'un Soc Zynq7000 ARM CortexA9 + FPGA


ZYBO
XILINX

3 Linux

Yocto est utilisé pour créer la distribution Linux.
https://www.yoctoproject.org/docs/current/mega-manual/mega-manual.html

3.1 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

3.2 Génération du bootloader FSBL

Le bootloader permet est lancé au démarrage de la carte. Il contient le logiciel de démarrage qui va permettre de configurer le système, comme les horloges internes, les entrées sorties du composant, et charger la couche matérielle avec le bitream du FPGA.
/home/nicolas/vivado/vivado/FPGA$_{x}ilinx/vivado2019_{2}/workspace/design_{1w}rapper_{c}orona2/export/desig$

/home/nicolas/vivado/vivado/FPGA_xilinx/vivado2019_2/workspace/dsfyoyo/Debug/dsfyoyo.elf

/home/nicolas/vivado/vivado/FPGA_xilinx/vivado2019_2/workspace/design$_{1w}rapper_{c}orona2/bitstream$

3.2.1 Génération boot.bin avec Xilinx Vivado

https://www.xilinx.com/support/documentation/sw$_{m}anuals/xilinx2019_{2}/ug1283-bootgen - user - guide.pdf$

nicolas@debianicolas: /yocto/lerobot/generation$_{b}oot$ more genere.bif
//arch = zynq; split = false; format = BIN
the$_{R}OM_{i}mage$:
[bootloader]u-boot-zybo-zynq7.elf
design$_{1w}rapper.bit$
nicolas@debianicolas: /yocto/lerobot/generation$_{b}oot$
/home/nicolas/Xilinx/Vivado/Vivado/2020.1/bin/bootgen -image genere.bif -o boute.bin -w

3.2.2 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)

3.3 Génération du boot avec le script genere.sh

Le script permet de générer le boot.bin à partir du fichier fsbl et du bitstream. Le bitstream contient la configuration matérielle du FPGA, aussi appelée "accélération matérielle".

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

3.4 Génération du Linux avec Yocto

Pour cloner yocto en local *git://git.yoctoproject.org/poky.git*

Il faut aussi cloner le repertoire meta-xilinx de la carte *https://github.com/Xilinx/meta-xilinx.git*

On clone aussi le répertoire java *git clone git://git.yoctoproject.org/meta-java*

Ajout dans le fichier bblayers.conf

```
/home/nicolas/yocto/poky/meta
/home/nicolas/yocto/poky/meta-poky
/home/nicolas/yocto/poky/meta-yocto-bsp
/home/nicolas/yocto/poky/meta-xilinx/meta-xilinx-bsp
/home/nicolas/yocto/poky/meta-xilinx/meta-java
/"
```

3.5 OpenDDS

https://opendds.org/

on ajoute la recipe dans yocto(presque) comme indiqué dans : *https://github.com/oci-labs/meta-opendds*

Pour compiler :

source /OpenDDS-3.15/setenv.sh
perl $MPC_ROOT/mpc.pl MessengerMinimal.mpc -type gnuace
make -f GNUmakefile.Messenger$_P$ublisher

3.6 Flashage de 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.

3.7 Génération bootgen avec le script genere.sh

Afin de copier l'image Linux générée pat Yocto, on commence par formatter la carte SD avec la commande :

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

3.8 Configuration des partitions

La carte SD a besoin de 2 partitions, créées avec la commande :

sudo fdisk /dev/mmcblk0

Utilisez fdisk comme indiquée ci-dessous :

```
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
```

3.9 création des systèmes de fichier

Il faut créer les systèmes de fichiers avec les commandes suivantes :

```
sudo mkfs.vfat -F 32 -n boot /dev/mmcblk0p1
sudo mkfs.ext4 -L root /dev/mmcblk0p2
```

3.10 Génération du Linux avec Yocto

3.10.1 Ajout des composants dans la distribution

Il faut ajouter les composants à la distribution de base

- *OpenDDS, qui permet d'échanger des données entre le Robot et la télécommande*
- *Git, permet de mettre à jour les logiciels sans avoir à recréer une distribution*

3.10.2 Génération de la distribution

Il faut générer la distribution Linux, et ensuite la claquer sur la carte SD
Si ce n'est pas fait,

- *cloner le projet dans un répertoire*

  git clone -b zeus git://git.yoctoproject.org/poky.git

- *Aller dans le répertoire poky*

  source oe-init-build-env

  Dans le fichier local.conf il faut rajouter :

- *ssh-server-openssh, afin de se connecter en ssh sur la cible*

- *tools-sdk, afin de compiler le logiciel sur la cible*

- *tools-debug, afin d'avoir les outils de deboguage (gdb) sur la cible*

```
Additional image features

EXTRA_IMAGE_FEATURES ?= "debug-tweaks"
EXTRA_IMAGE_FEATURES += "ssh-server-openssh"
EXTRA_IMAGE_FEATURES += "tools-sdk"
EXTRA_IMAGE_FEATURES += "tools-debug"
```

Maintenant on peut générer le linux :

- *Se mettre dans le repertoire poky et lancer source oe-init-build-env pour charger l'environnement de compilation*

- *core-image-full-cmdline pour générer le linux*

- *runqemu qemuxarm pour lancer l'émulateur. Attention à bien parametrer local.conf comme suit :*

```
# Additional image features
#
# This sets the default machine to be qemux86 if no other machine is
selected:
MACHINE ??= "qemuarm"
#MACHINE ??= "zybo-zynq7"
```

3.11 mettre le linux dans la carte SD

- *aller dans le répertoire : /home/nicolas/yocto/poky/build/tmp/deploy*

- *flasher la carte SD : 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/*
  *cp * /media/nicolas/boot/*

  sudo mount /dev/mmcblk0p2 /mnt/mmcblk0p2

- *compiler*
  /poky/meta-xilinx/ $ cd ..
  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"
- *Génération du boot.bin avec le bitstream*
  arch = zynq; split = false; format = BIN
  the_ROM_image:
  [bootloader]/home/nicolas/vivado/vivado/FPGA_xilinx/vivado2019_2/workspace/dsfyoyo/Debug/dsfyo
  /home/nicolas/vivado/vivado/FPGA_xilinx/vivado2019_2/workspace/dsfyoyo/_ide/bitstream/design_1_u
  /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

3.12 Lancement du logiciel au démarrage

*Pour lancer le robot au démarrage de Linux, il faut ajouter les lignes suivantes dans **/etc/inittab/** :*

```
1:12345:respawn:/sbin/getty 38400 tty1
nob:12345:once:su - nicolas -c "/home/nicolas/demarre_robot.sh"
```

4 Logiciel

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

4.1 Compilation du logiciel

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

sh compile_simone.sh
mise au point avec gdb
gdb sorie
(gdb) set args simuPC = lance en mode simulation pc

4.2 Architecture du logiciel

4.3 Execution du logiciel

mode simulation

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

Pour cela taper :

5 FPGA

Le materiel et les couches basses du logiciel sont gérées par la partie FPGA du composant. Le FPGA s'occupe de générer les signaux électriques (horloges, bus spi, contrôle des moteurs...) et les protocoles des différents capteurs (capteur de distance, gyroscope, micro ...). Le logiciel n'a plus qu'a récupérer les différents mesures en mémoire.

6 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 robt est établie, les boutons de pilotages apparaissent.