Environment Set-up

ELDK 3.1.1 installation

- only used to compile the kernel and u-boot. This version is mandatory as the wifi module is binary only
- get the iso here ftp://ftp.sunet.se/pub/Linux/distributions/eldk/3.1.1/arm-linux-x86/iso/arm-2005-03-06.iso
- `sudo mkdir -p /mnt/loop`
- `sudo mount -o loop arm-2005-03-06.iso /mnt/loop`
- `cd /mnt/loop`
- `sudo ./install -d /opt/eldk arm_trg`
- `sudo umount /mnt/loop`

buildroot installation

- Get the necessary packages:
  - fedora: `sudo yum install autoconf automake bison byacc flex gcc gcc-c++ make makeinfo ncurses-devel subversion texinfo zlib-devel`
  - ubuntu/debian:
    - `sudo apt-get install build-essential libncurses-dev bison flex texinfo zlib1g-dev gettext libssl-dev autoconf subversion wget`
    - change `/bin/sh` to `/bin/bash` instead of `/bin/dash`: `dpkg-reconfigure dash`, select no
  - get the source tarball on meccano site (sdk.tar.bz2) and untar it
  - `cd builroot-spykee`
  - `make defconfig` : select the default configuration for spykee
  - `make` : compile the crosstoolchain the first time you run it and the firmware

Generate a custom firmware

- `cd buildroot-spykee`
- `make menuconfig:` Add the software you want to use. Also check debug/no debug under "`Package Selection for the target`" if you want to add ssh the firmware. It is off by default.
- Make the modification you need to the source files.
- `make`
- this will create two files: `uImage` (linux kernel) et `rootfs.arm_nofpu.squashfs` (filesystem)
- get the script `do-firmware.py` in `buildroot-spykee/package/wavestorm/firmware-scripts` and

do `do-firmware.py buildroot-spykee`, this will take the linux kernel and the filesystem to make a firmware file that you can upload with the windows/mac software.

Sources layout

- Standard buildroot, taken from rev x
- 2 directories added:
  - `package/wavestorm :` contains the software that controls the robot behaviour
  - `target/device/spykee` : this has the filesystem tree and the configuration for various bits (linux kernel, busybox)
- in `package/wavestorm`:
  - `spykee`: communication with the 8051, battery charging, autoparking, configuration, misc. utilities
  - `robot`: handle the communication with the end-user software, camera, sound mixing
  - `scripts`: /etc/rc.d, startup scripts, wifi configuration
  - `firmware-scripts`: script to generate a firmware for the pc software
  - `base`: code of the 8051 in the charging station
  - `c8051f332`: code of 8051 on the main board
  - `ov530-driver`: driver for the camera board
  - `proprietary_modules`: binary linux kernel modules (only wifi)
  - `quickboot:` file related to power management
  - `spykee_prod_test`: pygtk programs used for manufacturing tests
- toolchain is in `build_arm_nofpu/staging_dir`, you can do `export PATH=$PATH:$(PATH_TO_BUILDROOT)/build_arm_nofpu/staging_dir/bin/` to get it into your path

Robot update example

- Make modifications the modifications in `package/wavestorm/spykee` for instance
- `rm -fr build_arm_nofpu/wavestorm build_arm_nofpu/root` to erase the previous object files
- `make` (or `make -s` to show only errors among the verbose output)

Mise à jour du numéro de version du firmware:

- file `target/device/spykee/target_skeleton/firmware-version`
- `rm -fr build_arm_nofpu/root`
- `make`
- `do-firmware.py .`

Update by ssh

- You can update the robot by ssh, this is slightly faster:
  - `scp rootfs.arm_nofpu.squashfs root@ip_robot:/tmp`
  - Credentials are `root/marvell`
- On the robot, `flashcp rootfs.arm_nofpu.squashfs /dev/mtd1`

- Max size of `rootfs` is `2818048` octets.
- `reboot`

busybox update

- `cd build_arm_nofpu/busybox-1.7.2`
- `make menuconfig`
- `cp .config ../../target/device/spykee/busybox.config`
- `cd ../..`
- `rm -fr build_arm_nofpu/busybox-1.7.2`
- `make`

add ssh

- Temporary: flash with the firmware '`firmware-dropbear.bin`', this won't update the robot but will launch ssh instead
- Create a firmware with ssh: enable the option "`package selection for the target`" => "`DEBUG flag for various packages`" with `make menuconfig`