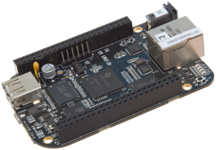
FreeBSD and Beaglebone Black, a robotic application.

Fabio Balzano
fabio.balzano@elfarolab.com

University of Ottawa,
Canada

May 17, 2014

The robot



BSDCan 2014
14-17 May
Ottawa, Canada

System description

What is this?

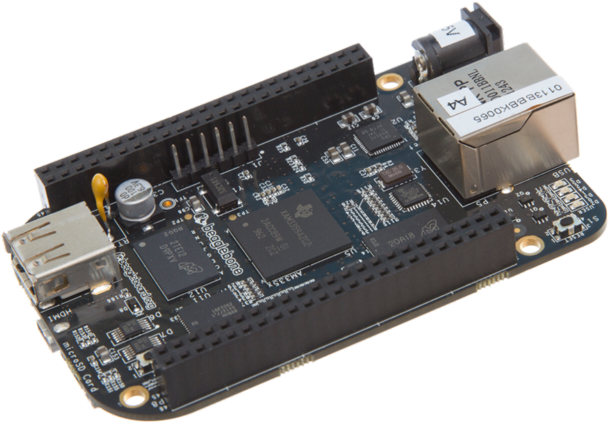
- it is a ROV - Remote Operated Vehicle
- it is based on Freebsd 11 embedded into an ARM board
- it provides an embedded web application via WIFI
- it provides real time video streaming
- it provides ultrasonic sensor for reading distances
- it provides I/O expansions for actuators

System description

Materials:

- Beaglebone Black + FreeBSD 11 current
- Protoboard for I/O
- WIFI antenna with USB interface
- Gimbal for the webcam and ultrasounds sensor
- Hobby RC car
- Battery
- Jumper wires and connectors

Beaglebone Black



Beaglebone Black

Features:

- 1GHz ARM Cortex A8
- 512MB DDR3 RAM
- 2Gbyte flash eMMC
- 4GB 8-bit eMMC on-board flash storage
- 3D graphics accelerator
- NEON floating-point accelerator
- 2x PRU 32-bit microcontrollers
- USB client and host ports
- Ethernet 10/100
- HDMI port
- 2x 46 pin headers

FreeBSD on BBB

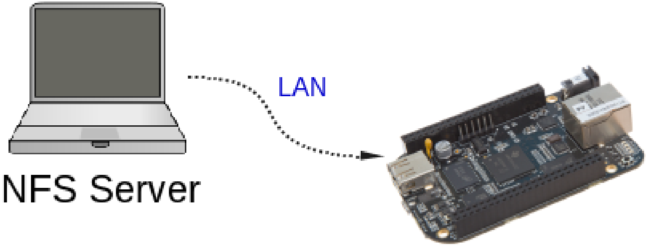
- It is stable
- Hardware resources are not wasted
- Almost everything is supported including the PRU
- Suitable for networking or data processing

FreeBSD and Beaglebone Black

STEPS:

- get the source of FreeBSD
- download the Crochet-FreeBSD script
- run the crochet script
- copy the produced image to a microSD card

Laboratory setup



Deployment server

Needed for:

- minimize the write cycles on the flash microSD card
- FreeBSD 11 microSD card image generation
- Host all the source code and packages

NFS exports:

- source dir of FreeBSD
- distfiles ports subdir
- packages ports subdir
- extra needed software

BSDCan 2014
14-17 May
Ottawa, Canada

Crochet-FreeBSD

Download it from:

- `https://github.com/kientzle/crochet-freebsd`

Crochet-FreeBSD

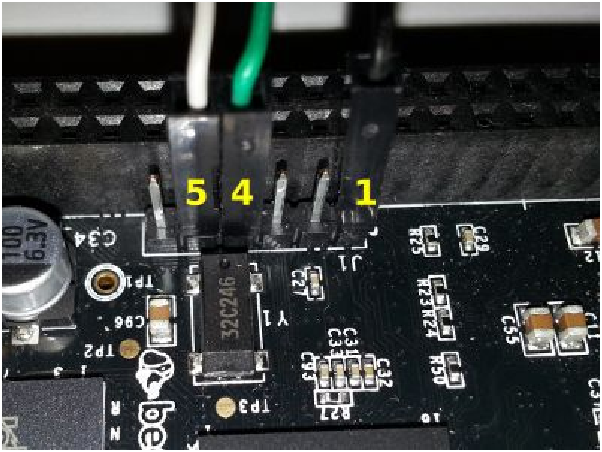
Building procedure:

```
# cd <crochet_path>
# cp config.sh.sample config.BBB.sh
# vi config.BBB.sh
# /crochet.sh -c config.BBB.sh

  coffe...

# dd if=<IMG> of=/dev/da0 bs=1m
```

Serial console cable

Serial console cable

TX - RX - GND



Serial console terminal

Three common ways to connect:

```
# cu -s 115200 -l /dev/ttyU0

# screen /dev/ttyU0 115200

# minicom
```

Flattened device tree - fdt

Benefits:

- Put outside the kernel the device definitions
- Make it easier to enable/disable devices after a kernel build.
- Same kernel for multiple similar boards

At the boot, U-boot load .dtb into the memory

The kernel pickup the keys and load the drivers

Flattened device tree - fdt

It is used for:

- To describe the hardware
- To list all the devices and their properties
- To enable or disable driver and devices
- To mux the pins for alternatives functions

Do you need to enable or redefine some I/O pins?

- Edit the .dts clear text file
- Compile the .dts into a .dtb binary file with dtc

Laboratory

My setup and results

My setup

Disabled unneeded stuff into the kernel config

Disabled all the debugging including WITNESS

I wanted to try a patch set to rise the CPU clock to 1GHz

I wanted to maximize the performance for the video processing

1Ghz patches

Default clock frequency is 500Mhz

Patched and used the new u-boot 2014.01

I used the patches from Xuebing Wang

Result:

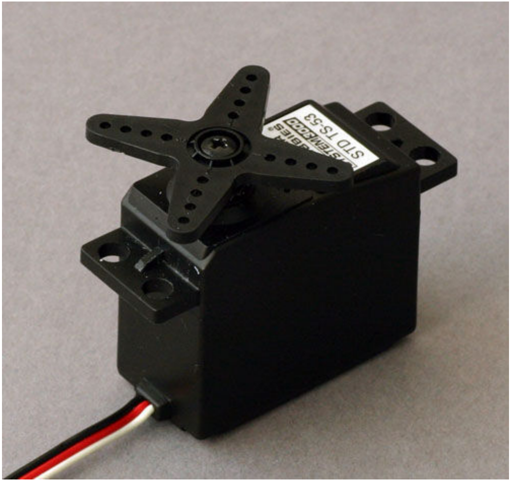
the kernel is running at 1 GHz, the CPU is a little bit warmer, an heat sink should be installed before real heavy processing.

Better solution:

- a FreeBSD driver that expose a read/write sysctl to reprogram the CPU frequency

BBB PWM pins

The robot has 4 servo motors:



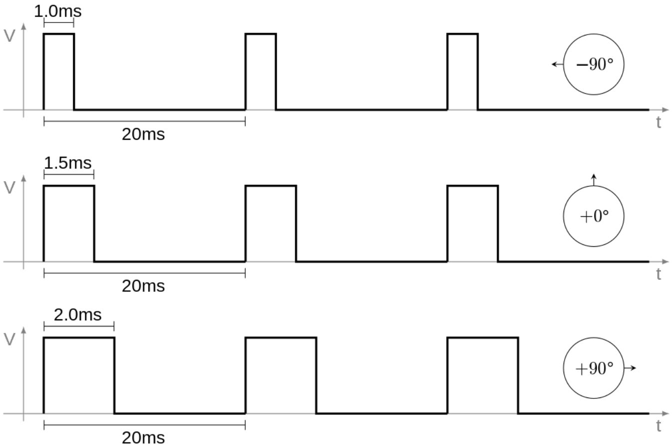
BSDCan
2014
14-17 May
Ottawa, Canada

4 servo motors

- 2 for the camera Gimbal
- 1 for the steering
- 1 for the ESC

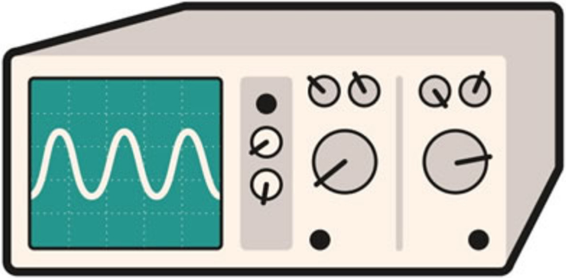
PWM signal

The width of the pulse drive the servo:



PWM signal from the BBB pins

Test of the signals



Test of the PWM signal

```
# sysctl dev.am335x_pwm.1.period=1500
# sysctl dev.am335x_pwm.1.dutyA=300
result frequency: 66 KHz  ->should be 666 KHz
length of period: 15 us ->should be 1.5 us
length of pulse: 2 us

# sysctl dev.am335x_pwm.1.period=1500000
# sysctl dev.am335x_pwm.1.dutyA=10000
result frequency: 1.71 KHz -> should be 666Hz
length of period:  585 us
length of pulse:  100 us

# sysctl dev.am335x_pwm.1.period=1800000
# sysctl dev.am335x_pwm.1.dutyA=10000
result frequency: 3.24 KHz -> should be 555Hz
length of period:  308 us
length of pulse:  100 us
```

BBB PWM pins

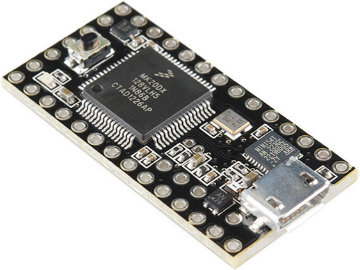
Actual minimum frequency was 1.5KHz

further work to expand the range and correct the configuration keys

Help please!

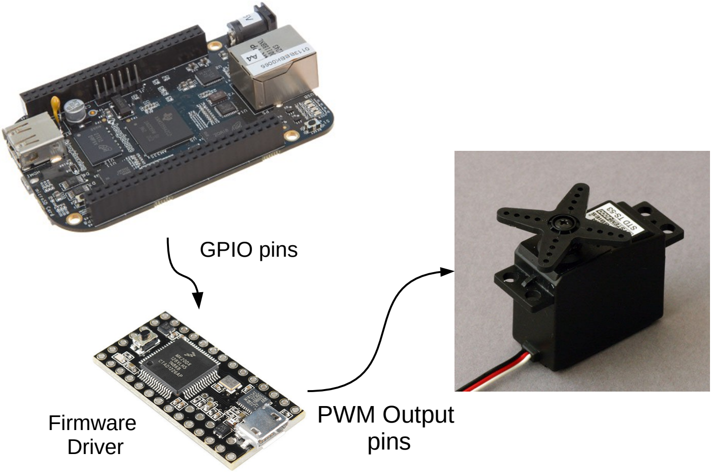
Alternative to the BBB PWM pins

Teensy 3.0 board:



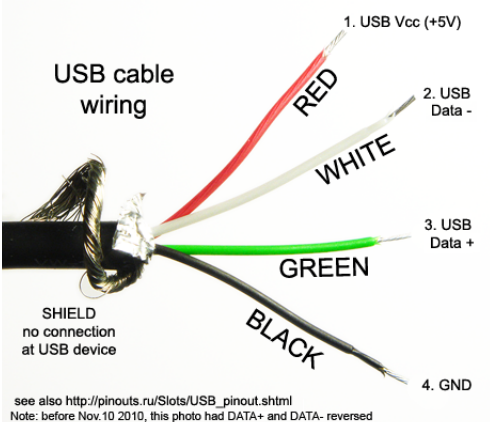
BSDCan
2014
14-17 May
Ottawa, Canada

Teensy to drive the servo motors



USB Hub

- BBB has only one host USB port
- Cut the power supply wires of the USB cable



USB Hub

- USB hotplug is working
- some problems with the video frames transmission

WLAN link

- To pilot the robot is via WIFI
- The system is configured as an access point with hostapd
- The user can pilot the robot using a web application installed on the BBB

Serverside

- Python Flask web application with websockets
- c code to drive the GPIO pins for the servo motors
- c code to read the distance sensors
- OpenCV realtime video processing

Improvements

- kernel drive to increase CPU frequency
- better implementation of the PWM output signal
- further development with OpenCV
- ...

Demo

DEMO!

BSDCan 2014
14-17 May
Ottawa, Canada

Repository:

- https://bitbucket.org/fabiodive/bsdcan2014

My email address:

- fabio.balzano@elfarolab.com

Thank you for your attention!

flood me questions :-)