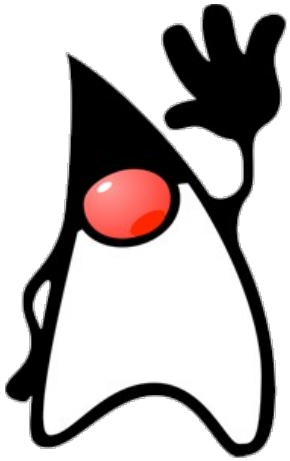
beaglebone

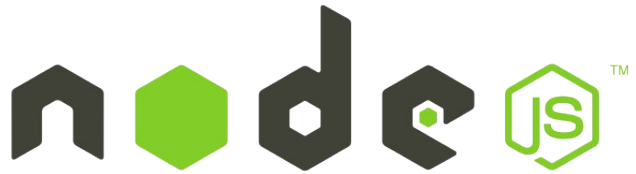
- Arduino^10 + Mix de technos !

Laurent HUET
SOFTEAM

Plan

- **Généralités**
  - Hardware
  - Software
- **Labs**
  - Entrées/Sorties numériques
  - Bus 1-wire avec GPIO
  - Entrée analogique
  - Bus I2C
  - ~~Port série (UART)~~
  - Mashup !





Qui suis-je ?

@lhuet35

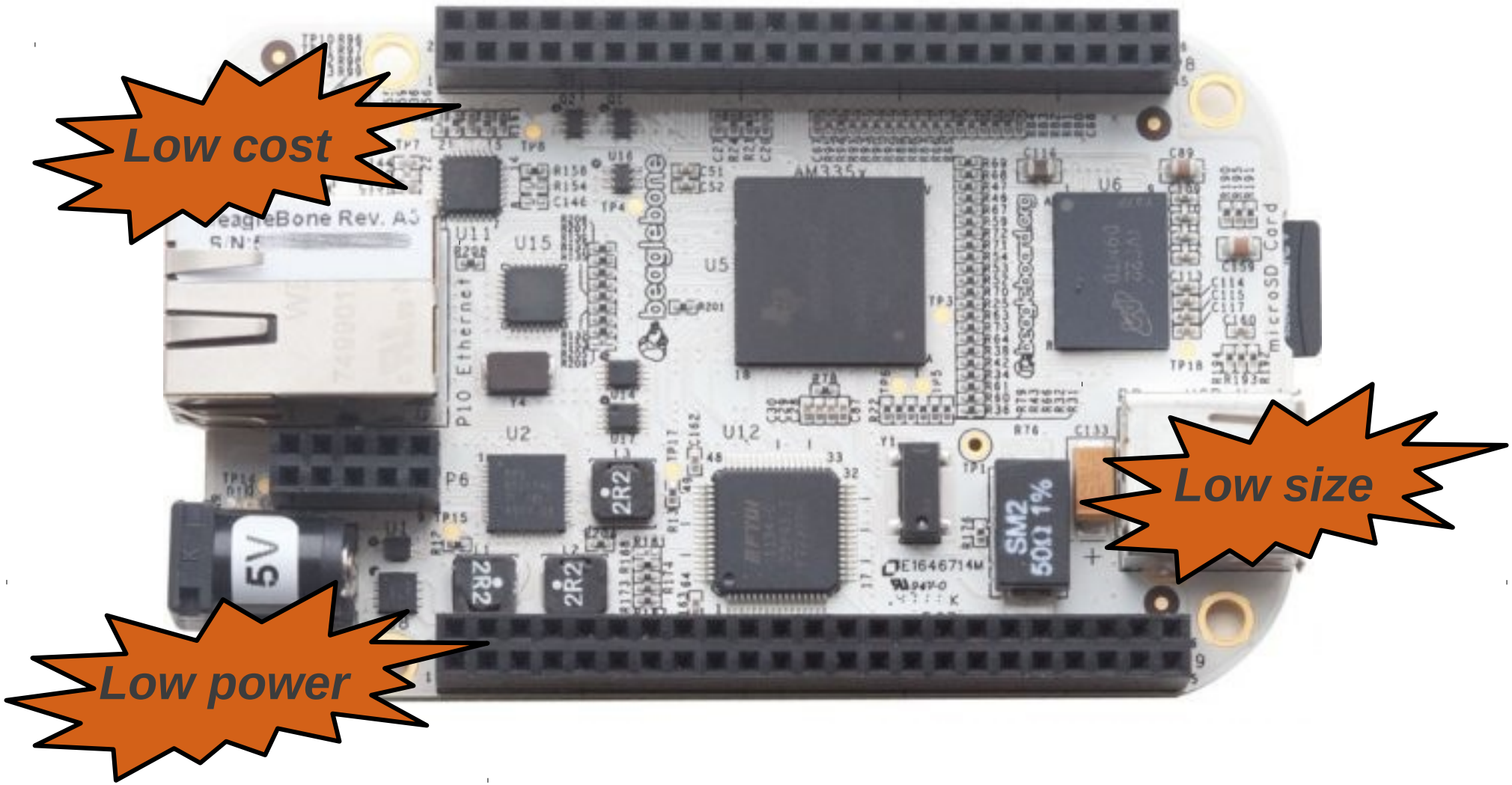
laurent.huet { @softeam.fr (pro) / @gmail.com (perso)

- **Softeam depuis 2004**
  - *Consultant / Formateur / Architecte JavaEE*
  - *Responsable Technique Softeam Ouest*
- **Sema / SchlumbergerSema / Atos Origin - 2000 à 2004**
- **S3EB (filiale Bouygues) – 1996 à 2000**
  - GTB / GTC (Gestion Technique du Bâtiment / Centralisée)

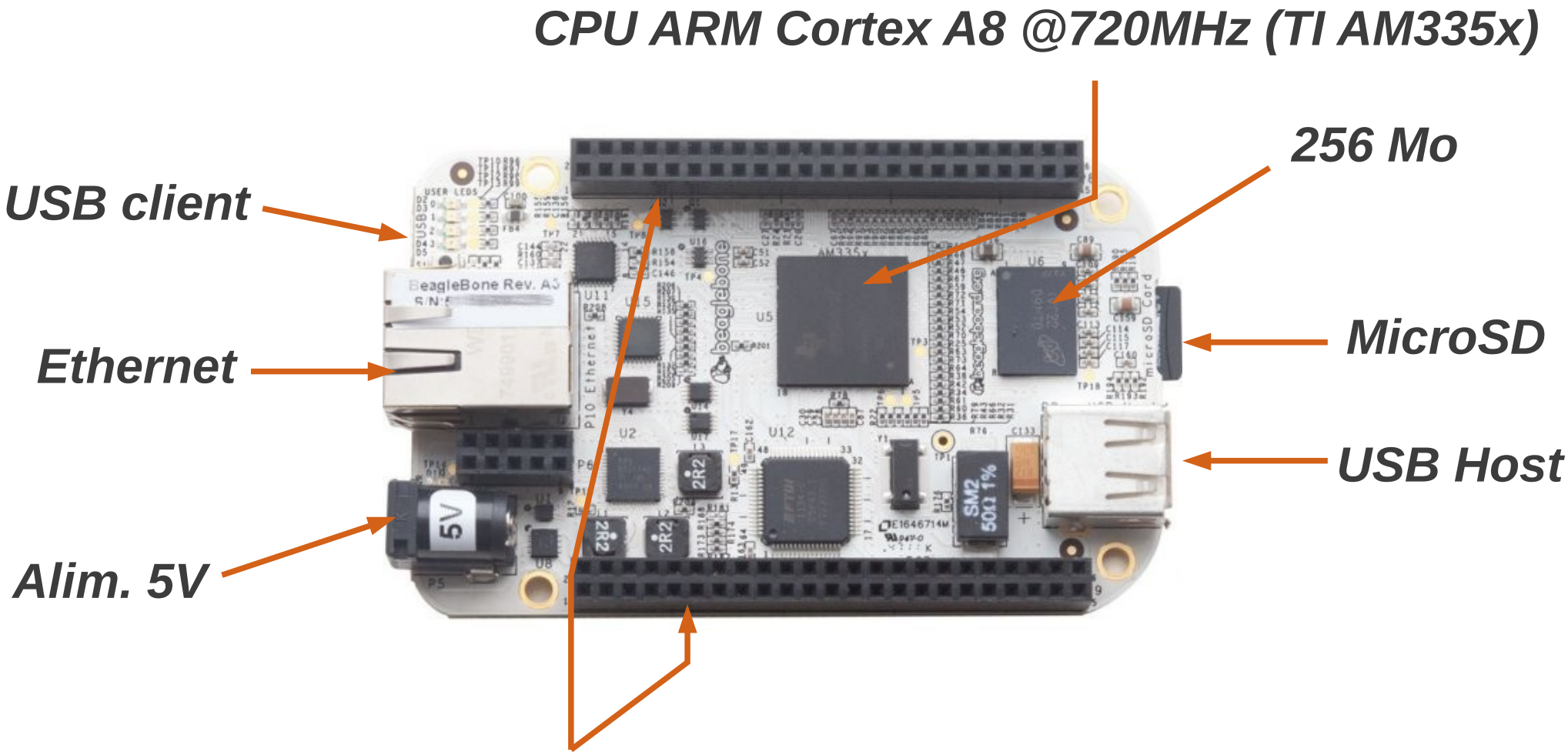
- C'est quoi ?
- Ca sert à quoi ?

Carte ARM « Open source hardware »

http://beagleboard.org/bone



Caractéristiques



Cartes d'extension possible (capes)

DVI Cape

LCD Cape

Proto Cape

Liste de cartes d'extension maintenue par les créateurs de la beaglebone
http://circuitco.com/support/index.php?title=BeagleBone_Capes:

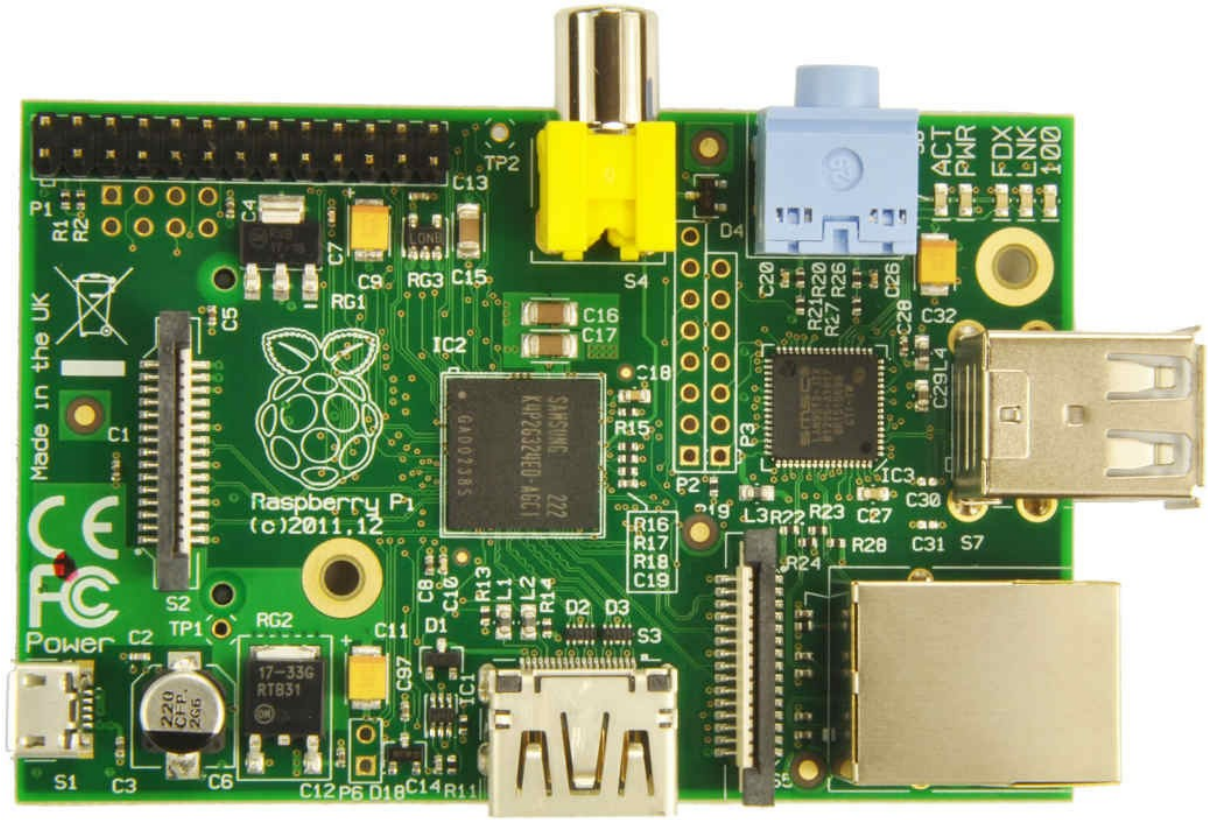
Autres cartes ...

Arduino(s) - http://arduino.cc/

- **Microcontrolleurs**
  - Programme limité (4ko à 128ko selon les versions)
  - Pas d'OS – Programme en “pseudo-C” en mémoire flash
  - ~16 MHz !

Raspberry Pi - http://www.raspberrypi.org/

- **Carte « Low cost »**
  - CPU ARMv6 @700MHz – 128 ou 256 Mo RAM
  - “Linux inside”
  - Connectique :
    - SD Card
    - HDMI + RCA Vidéo
    - Son (jack 3.5)
    - USB Host
    - Lan RJ45
    - Connecteur 26-Pins
      - GPIO / I2C / SPI / UART



A quoi sert une Beaglebone ?

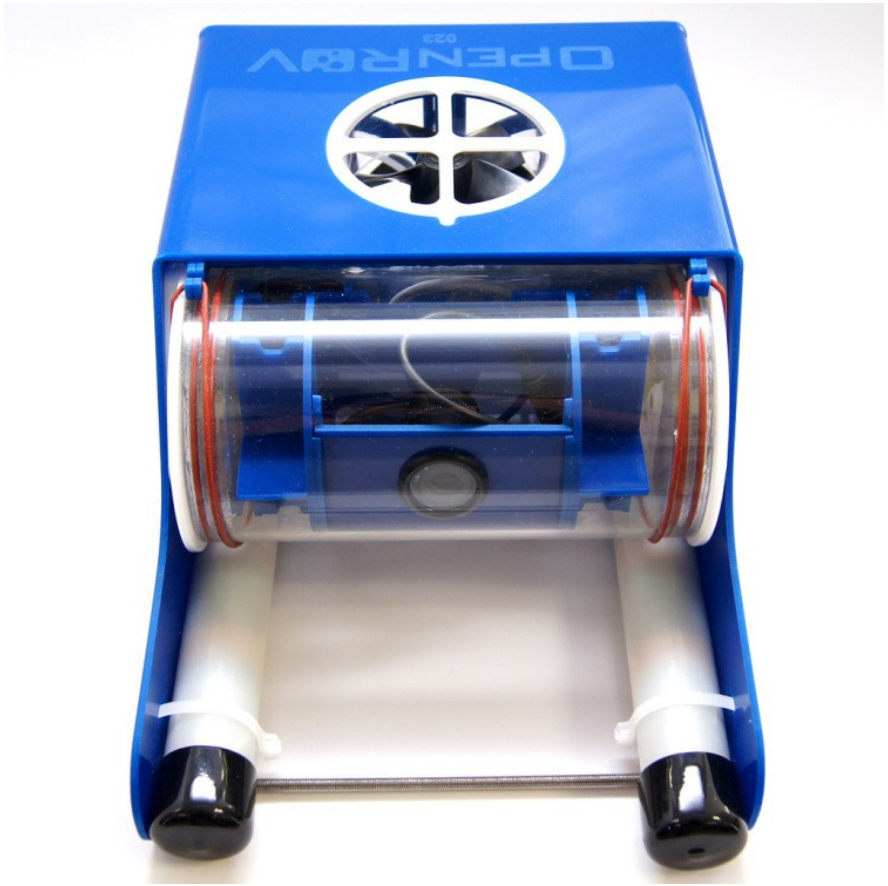
Quoi faire avec la carte Beaglebone ?

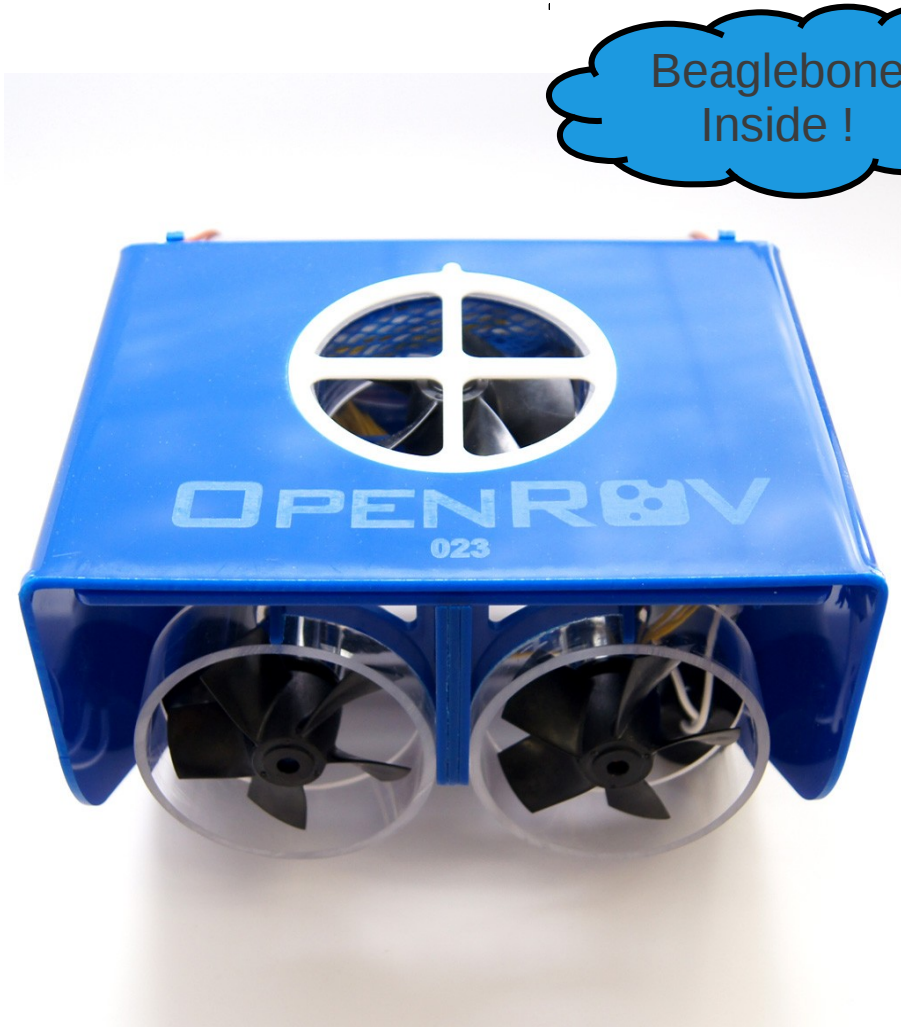
- **Station météo**
- **Domotique**
- **Robotique**
- **OpenCV et OpenNI**
  - Traitement de l'image en temps réel
  - Reconnaissance de la voix, mouvement, …
- **« Home Media Server »**

=> Multitude de projets existants :
http://beagleboard.org/project

OpenROV – http://openrov.com

- **« Open-source underwater robot for exploration and education »**





Quels OS sont supportés ?

- **Linux**
  - Ångström **-** *http://www.angstrom-distribution.org*
  - Ubuntu / Debian - *https://wiki.ubuntu.com/ARM/OMAP*
  - …

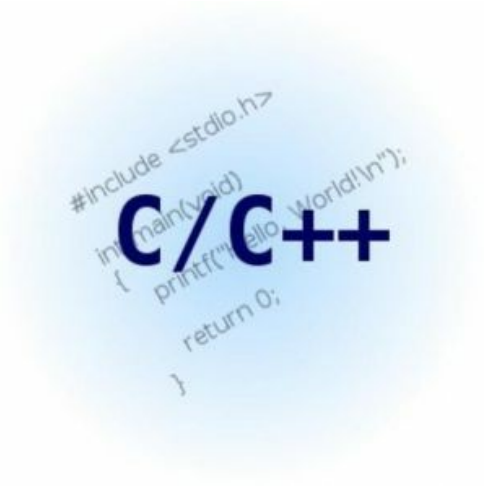
- **Android ICS & Gingerbread -** *http://www.ti.com/sitara-android*
- **Android Jelly Bean -** *https://code.google.com/p/rowboat/*

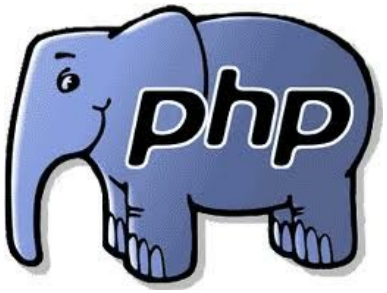
- ***Windows Embedded Compact 7***
  - *http://www.adeneo-embedded.com/*

- **Autres (BSD, ...)**

Quel langage utiliser ?

- **Celui qui vous convient !**





Partie Labs

- Les mains dans le cambouis ;-)

« Disclaimer »

- **DANGER**
  - Toujours vérifier les tensions !
  - Destruction de la carte possible
- **Tensions à respecter**
  - GPIO : 3,3 V / AIN : 1,8 V Max
- **Puissances à respecter**

Current	Name	P9			Name	Current
	GND	1		2	GND	
250mA	VDD_3V3EXP	3		4	VDD_3V3EXP	250mA
1000mA	VDD_5V	5		6	VDD_5V	1000mA
250mA	SYS_5V	7		8	SYS_5V	250mA
		:		:		
	GND	43		44	GND	
	GND	45		46	GND	

OS choisi pour la partie Labs

- **Distribution Ångström**
- **Avantages**
  - Livrée avec la microSD
  - Facilité de prise en main
    - Réseau via USB sans paramétrage
  - Noyau à jour
    - Patchs spécifiques beaglebone
    - Modules I2C / 1-Wire / SPI / … par défaut
  - Fourni avec Cloud9 / nodeJS

« Pin muxing » - kesako

- Comment gérer la pénurie de connecteurs

Beaucoup de fonctionnalités

- 66 entrées/sorties numériques (GPIO)
- 7 entrées analogiques (AIN)
- 5 ports série (UART)
- 2 ports I2C
- 2 ports SPI
- 2 bus CAN
- Autres fonctions (LCD, PWM, ...)

Connectique limitée : Pas assez de Pins !

- **92 pins disponibles (2x46)**
- **66 GPIO + 5 UARTs + 7 AIN + 2 ports I2C + 2 ports SPI + 2 bus CAN + … > 92 pins !**
- **Solution : « Pin muxing »**
  Configuration logicielle des pins

=> Toutes les fonctions ne peuvent pas être utilisées en même temps

« Pin Muxing » : Réponse à la pénurie !

- **1 Pin = plusieurs fonctions**
  - Ex. : GPIO ou UART.Rx ou LCD_datax ou …
- **8 modes possibles (Cf. Manuel de référence)**
  - Tableau des fonctions des Mode0 à Mode7 par Pin
- **Mode choisi par logiciel**
  - (pilotage d'un registre interne du CPU)

Entrées/Sorties numériques

- GPIO (General Perpose Input/Output)
- Exemples
  - Allumer une lampe (led)
  - Détection d'un bouton

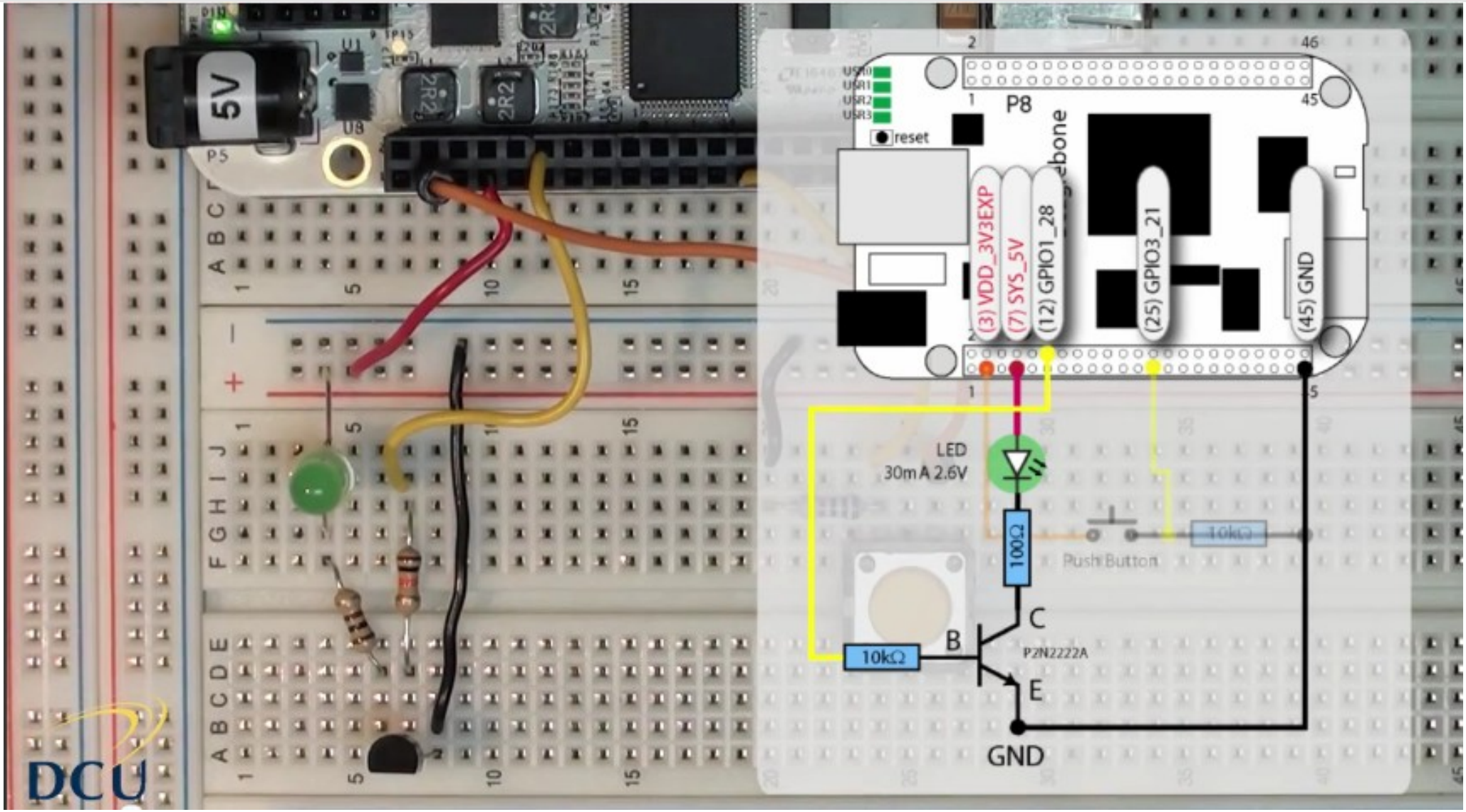
Comment utiliser les GPIO ?

- **API du noyau**
  - Code C / C++ (`gpio.h`)
- **Utilisation d'un système de fichier virtuel (Sysfs)**
  - API = lecture/écriture de fichiers !
  - Principes
    - Activer le port GPIO
    - Configurer le port en input / output
    - Lire (si input) ou écrire (si output) dans le fichier
    - Désactiver le port GPIO

Montage 1 – Sortie numérique

- Pilotage d'une sortie numérique (allumer une led)

Montage utilisé – Sortie digitale



Source : http://www.youtube.com/user/DerekMolloyDCU

Identification du pin

- **Brochage du connecteur P9 (cf. Manuel)**

Table 11. Expansion Header P9 Pinout

SIGNAL NAME	PIN	CONN		PIN	SIGNAL NAME
	GND	1	2	GND	
	VDD_3V3EXP	3	4	VDD_3V3EXP	
	VDD_5V	5	6	VDD_5V	
	SYS_5V	7	8	SYS_5V	
PWR_BUT*		9	10	A10	SYS_RESETn
UART4_RXD	T17	11	12	U18	GPIO1_28
UART4_TXD	U17	13	14	U14	EHRPWM1A

=> GPIO1_28 sur Pin 12 / connecteur P9

« Pin muxing »

- Fonction GPIO pour le pin P9_12

Identification du fichier de configuration

- **Fichiers de configuration**

  `/sys/kernel/debug/omap_mux/*`

- **Convention : Nom du fichier = nom du Mode0**

```
lhuet@lhuet-Laptop: ~
root@beaglebone:~# ls /sys/kernel/debug/omap_mux/
ain0                gpmc_a4     gpmc_ad6        lcd_ac_bias_en  lcd_pclk        mii1_rxd3     spi0_d0
ain1                gpmc_a5     gpmc_ad7        lcd_data0       lcd_vsync       mii1_rxdv     spi0_d1
ain2                gpmc_a6     gpmc_ad8        lcd_data1       mcasp0_aclkr    mii1_rxerr    spi0_sclk
ain3                gpmc_a7     gpmc_ad9        lcd_data10      mcasp0_aclkx    mii1_txclk    uart0_ctsn
ain4                gpmc_a8     gpmc_advn_ale   lcd_data11      mcasp0_ahclkr   mii1_txd0     uart0_rtsn
ain5                gpmc_a9     gpmc_ben0_cle   lcd_data12      mcasp0_ahclkx   mii1_txd1     uart0_rxd
ain6                gpmc_ad0    gpmc_ben1       lcd_data13      mcasp0_axr0     mii1_txd2     uart0_txd
ain7                gpmc_ad1    gpmc_clk        lcd_data14      mcasp0_axr1     mii1_txd3     uart1_ctsn
board               gpmc_ad10   gpmc_csn0       lcd_data15      mcasp0_fsr      mii1_txen     uart1_rtsn
ecap0_in_pwm0_out   gpmc_ad11   gpmc_csn1       lcd_data2       mcasp0_fsx      mmc0_clk      uart1_rxd
emu0                gpmc_ad12   gpmc_csn2       lcd_data3       mdio_clk        mmc0_cmd      uart1_txd
emu1                gpmc_ad13   gpmc_csn3       lcd_data4       mdio_data       mmc0_dat0     usb0_drvvbus
gpmc_a0             gpmc_ad14   gpmc_oen_ren    lcd_data5       mii1_col        mmc0_dat1     usb1_drvvbus
gpmc_a1             gpmc_ad15   gpmc_wait0      lcd_data6       mii1_crs        mmc0_dat2     vrefn
gpmc_a10            gpmc_ad2    gpmc_wen        lcd_data7       mii1_rxclk      mmc0_dat3     vrefp
gpmc_a11            gpmc_ad3    gpmc_wpn        lcd_data8       mii1_rxd0       rmii1_refclk  xdma_event_intr0
gpmc_a2             gpmc_ad4    i2c0_scl        lcd_data9       mii1_rxd1       spi0_cs0      xdma_event_intr1
gpmc_a3             gpmc_ad5    i2c0_sda        lcd_hsync       mii1_rxd2       spi0_cs1
root@beaglebone:~#
```

Identification du fichier de configuration

- **Fichier de configuration pour P9_12**

  `/sys/kernel/debug/omap_mux/gpmc_ben1`

Table 12. P9 Mux Options Modes 0-3

PIN	PROC	SIGNAL NAME	MODE0	MODE1	MODE2	MODE3
1		GND				
2		GND				
3		DC_3.3V				
11	T17	UART4_RXD	gpmc_wait0	mii2_crs	gpmc_csn4	rmii2_crs_dv
12	U18	GPIO1_28	gpmc_be1n	mii2_col	gpmc_csn6	mmc2_dat3
13	U17	UART4_TXD	gpmc_wpn	mii2_rxerr	gpmc_csn5	rmii2_rxerr

Contenu du fichier de configuration

Contenu du fichier gpmc_ben1 => gpio1[28] = Mode7

```
# cat /sys/kernel/debug/omap_mux/gpmc_ben1
name: gpmc_ben1.gpio1_28 (0x44e10878/0x878 = 0x0037), b NA, t NA
mode: OMAP_PIN_OUTPUT | OMAP_MUX_MODE7
signals: gpmc_ben1 | mii2_col | NA | mmc2_dat3 | NA | NA |
                                                    mcasp0_aclkr | gpio1_28
```

Table 13. P9 Mux Options Modes 4-7

PIN	PROC	SIGNAL NAME	MODE4	MODE5	MODE6	MODE7
1		GND				
2		GND				
3		DC_3.3V				
11	T17	UART4_RXD	mmc1_sdcd		uart4_rxd_mux2	gpio0[30]
12	U18	GPIO1_28	gpmc_dir		mcasp0_aclkr_mux3	gpio1[28]

Configuration du pin

Ecriture du mode dans le fichier

```
# echo 7 > /sys/kernel/debug/omap_mux/gpmc_ben1

# cat /sys/kernel/debug/omap_mux/gpmc_ben1
name: gpmc_ben1.gpio1_28 (0x44e10878/0x878 = 0x0007), b NA, t NA
mode: OMAP_PIN_OUTPUT | OMAP_MUX_MODE7
signals: gpmc_ben1 | mii2_col | NA | mmc2_dat3 | NA | NA |      ↵
                                                  mcasp0_aclkr | gpio1_28
```

Utilisation de l'API Sysfs

- Il est temps d'allumer la lumière !

Export/Activation du port GPIO (sysfs)

- **Nom : gpio1[28]**
  - 1 = N° de “bank”
  - 28 = N° Pin
- **N° de GPIO sous Linux = (32 * N° bank) + N° Pin**
  - Gpio1[28] => (32 * 1) + 28 = ***60***
- **Export**
  - `echo 60 > /sys/class/gpio/export`

    => Mise à dispo du répertoire :

    `/sys/class/gpio/gpio60`

Utilisation du port GPIO

- **Configuration en sortie : fichier `direction`**

```
echo out > /sys/class/gpio/gpio60/direction
```

- **Fichier `value` pour activer la sortie**
  - Activation

```
echo 1 > /sys/class/gpio/gpio60/value
```

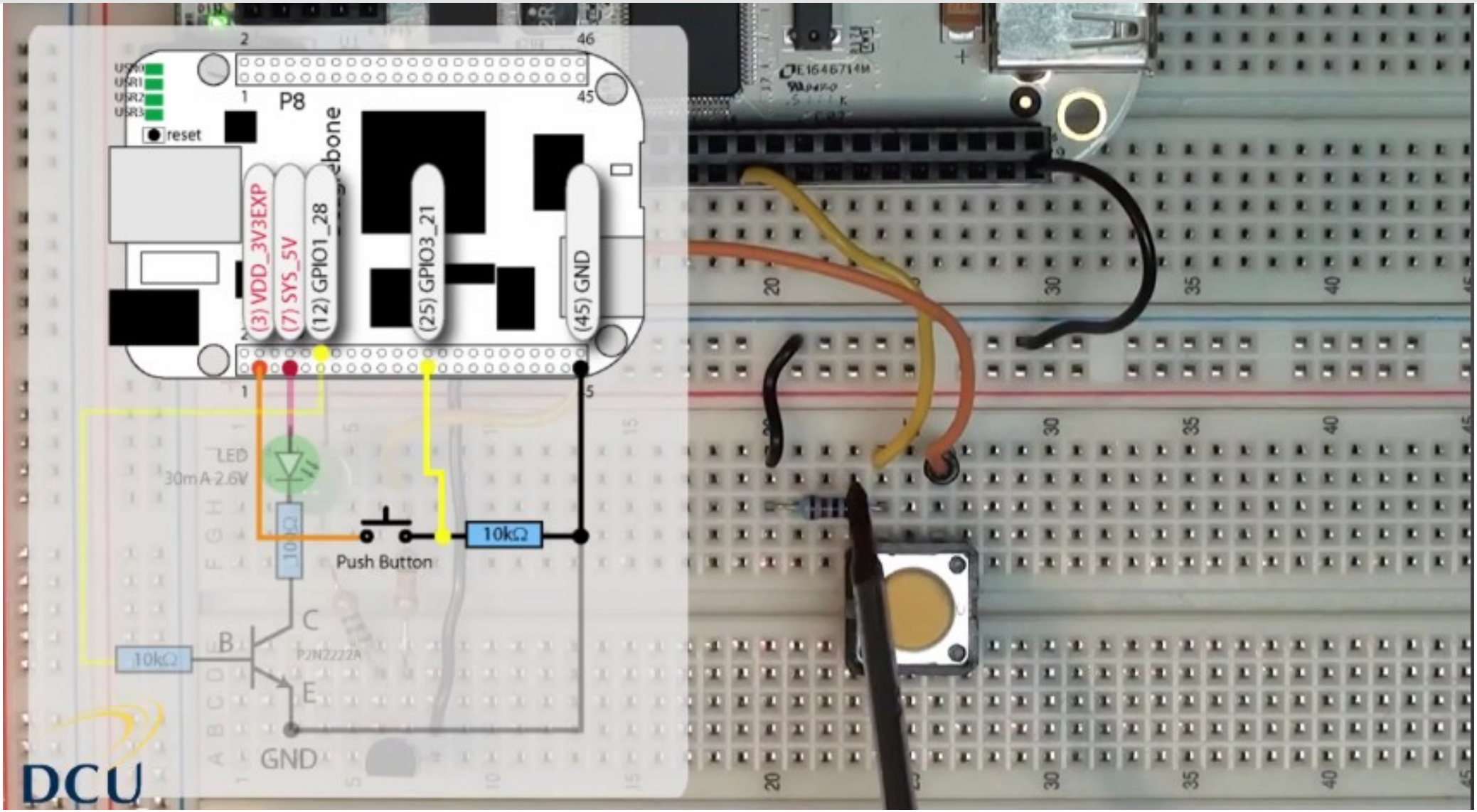
  - Désactivation

```
echo 0 > /sys/class/gpio/gpio60/value
```

Montage 2 – Entrée numérique

- Détection d'une fermeture d'un contact

Montage utilisé - Entrée digitale



Source : http://www.youtube.com/user/DerekMolloyDCU

« Pin muxing »

- Fonction GPIO en entrée pour le pin P9_25

Identification du mode à activer

- **Pin P9_25 : Nom du fichier (Mode0) : `mcasp0_ahclkx`**

			Mode 0	Mode 1	Mode 2	Mode 3
25	A14	GPIO3_21	mcasp0_ahclkx	eQEP0_strobe	mcasp0_axr3	mcasp1_axr1

- **Gpio3[21] => Mode7**

			Mode 4	Mode 5	Mode 6	Mode 7
25	A14	GPIO3_21	EMU4_mux2			gpio3[21]

Petite subtilité ...

- **Chaîne hexadécimale particulière pour le mode Input**

Bit	5	4	3	2	1	0
Set (1)	Input	Pull Up	Pull Enabled	Mode		
Clear (0)	Output	Pull Down	Pull disabled			

- **Bit 5 « enabled » pour le mode Input**

  => `0010 0111 = 0x27`

- **Configuration GPIO3_21**

```
echo 27 > /sys/kernel/debug/omap_mux/mcasp0_ahclkx
```

Utilisation de l'API Sysfs

- Créons notre interrupteur !

Utilisation du port GPIO3_21

- **Activation**
  - N° gpio = 3*32 + 21 = **117**

```
echo 117 > /sys/class/gpio/export
```

- **Configuration en input**

```
echo in > /sys/class/gpio/gpio117/direction
```

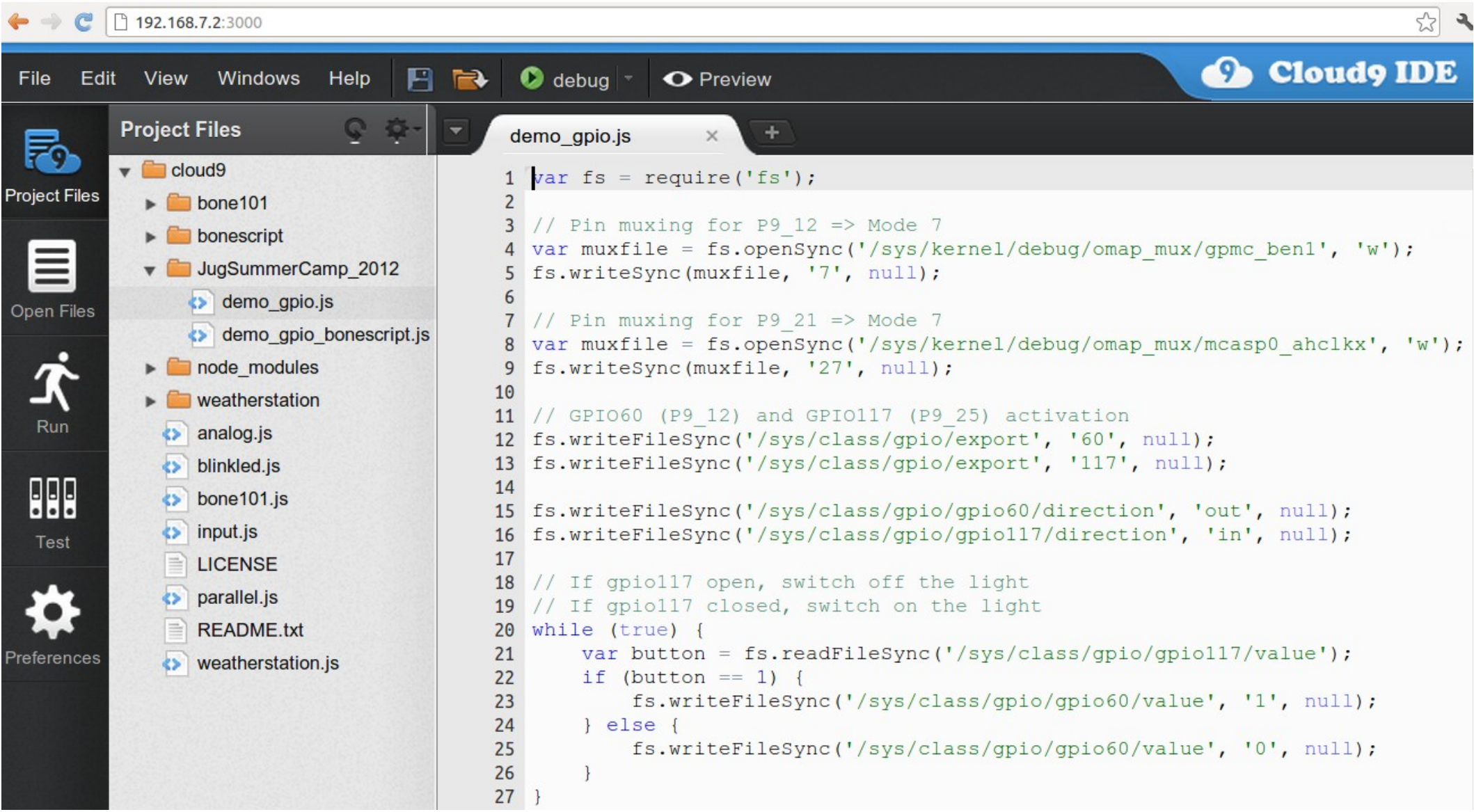
- **Lecture de l'état**

```
cat /sys/class/gpio/gpio117/value
```

- **Désactivation**

```
echo 117 > /sys/class/gpio/unexport
```

Node.js / Cloud 9



NodeJS avec librairie « bonescript »

demo_gpio_bones...

```
1  var bb = require('../bonescript');
2
3  var ledPin = bone.P9_12;
4  var buttonPin = bone.P9_25;
5
6  setup = function() {
7      pinMode(ledPin, OUTPUT);
8      pinMode(buttonPin, INPUT);
9  };
10
11 loop = function() {
12
13     var buttonState = digitalRead(buttonPin);
14     if (buttonState==1) {
15         digitalWrite(ledPin, HIGH);
16     } else {
17         digitalWrite(ledPin, LOW);
18     }
19 };
20
21 bb.run();
```

Exemple avec Python

- **Librairie utilisée :** https://github.com/alexanderhiam/PyBBIO

```
from bbio import *

buttonPin=GPIO3_21
ledPin=GPIO1_28

def setup():
    pinMode(ledPin, OUTPUT)
    pinMode(buttonPin, INPUT)
    digitalWrite(ledPin, LOW)

def loop():
    if digitalRead(buttonPin)==1:
            digitalWrite(ledPin, HIGH)
    else:
            digitalWrite(ledPin, LOW)

run(setup, loop)
```

Montage 3 - Bus 1-wire

- Est-ce qu'il fait chaud à La Rochelle ?

Bus 1-wire

- **1 fil « data »**
  - Alimentation 3 à 5V généralement utilisé
  - Longueur max : 20 m (environ)
- **Plusieurs composants sur un bus**
  - Topologie : Etoiles / Série / Parallèle
- **Capteurs divers**
  - Température (DS18B20)
  - Compteur d'impulsion (DS2423)
  - Intérupteur / Détecteur d'état (DS2405)
  - ...

1-wire sous Linux : module w1-gpio

- **Module w1-gpio intégré**
  - Implémente le protocole 1-wire
  - Inclus et configuré dans la distribution Ångström
  - Scanne le bus (autoprobe)
  - Mise à disposition de fichiers virtuels

- **Quel pin utiliser ?**

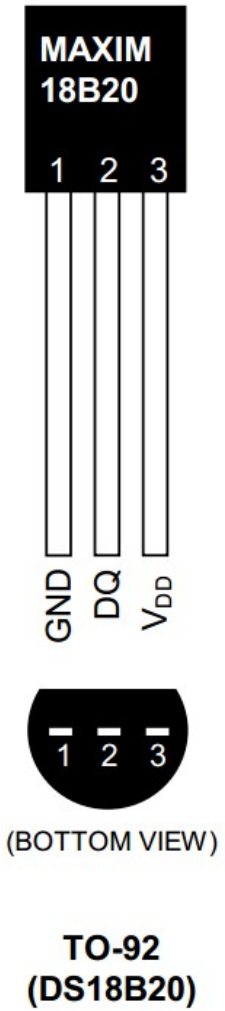
```
# dmesg |grep w1
[    0.229369] w1-gpio connected to P8_6
```

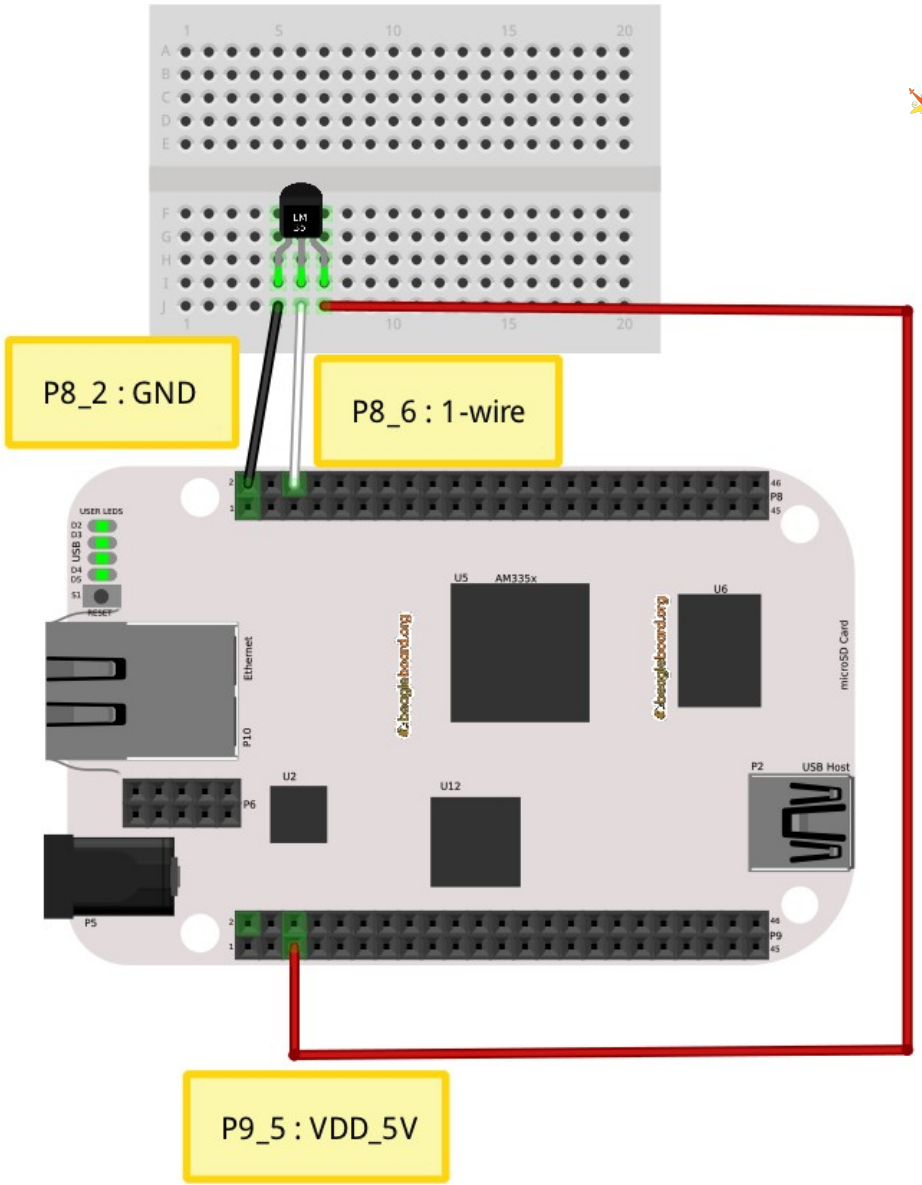
=> Connecteur P8 – Pin 6

Capteur de température

- **DS18B20 : plusieurs formats**

Mise en œuvre d'un DS18B20





- **Cablage**
  - Data sur P8_6
  - GND / 5V sur P8_2 / P9_5

Lecture d'une température

- **Détection automatique des composants sur le bus**
  - Répertoire `/sys/bus/w1/devices` mis à jour

```
root@beaglebone:/sys/bus/w1/devices# ls
28-000003a5aacc  w1_bus_master1
```

Nouveau répertoire pour le capteur DS18B20

- **Lecture du fichier `w1_slave`**
  - Accès direct au bus => lent

```
# cat 28-000003a5aacc/w1_slave
55 01 4b 46 7f ff 0b 10 d0 : crc=d0 YES
55 01 4b 46 7f ff 0b 10 d0 t=21312
```

Lecture de la température en Groovy

```
#!/usr/bin/env groovy

def f = "/sys/bus/w1/devices/28-000003c1b6e7/w1_slave"

def w1Sensor = new File(f).readLines()[1]
                          .split("t=")[1]

def sensorValue = (w1Sensor as int)/1000

println("Au JugSummerCamp, il fait super chaud : "
                                    + sensorValue + " °C")
```

LectureTemperature.groovy

Entrée analogique

- Exemple : Capteur de température

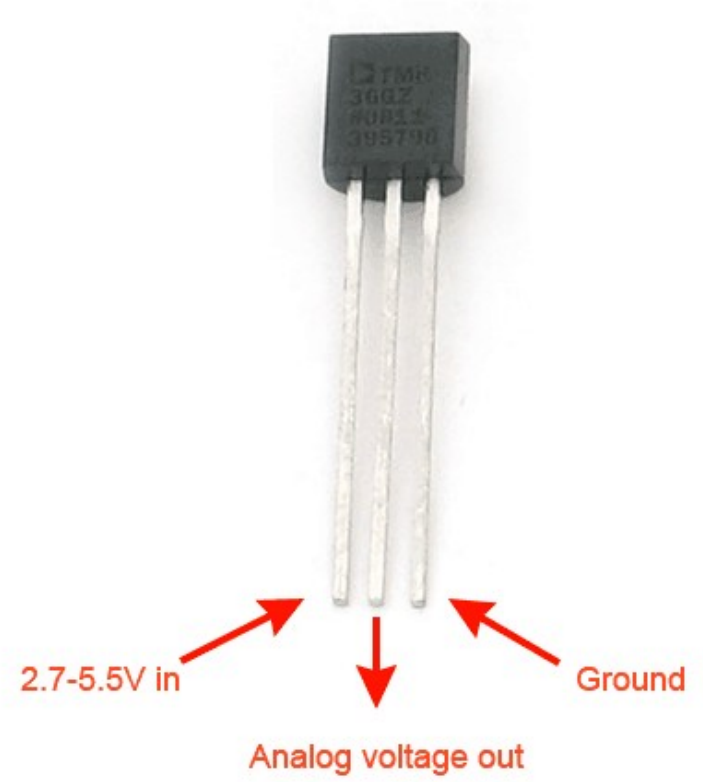
7 entrées analogies

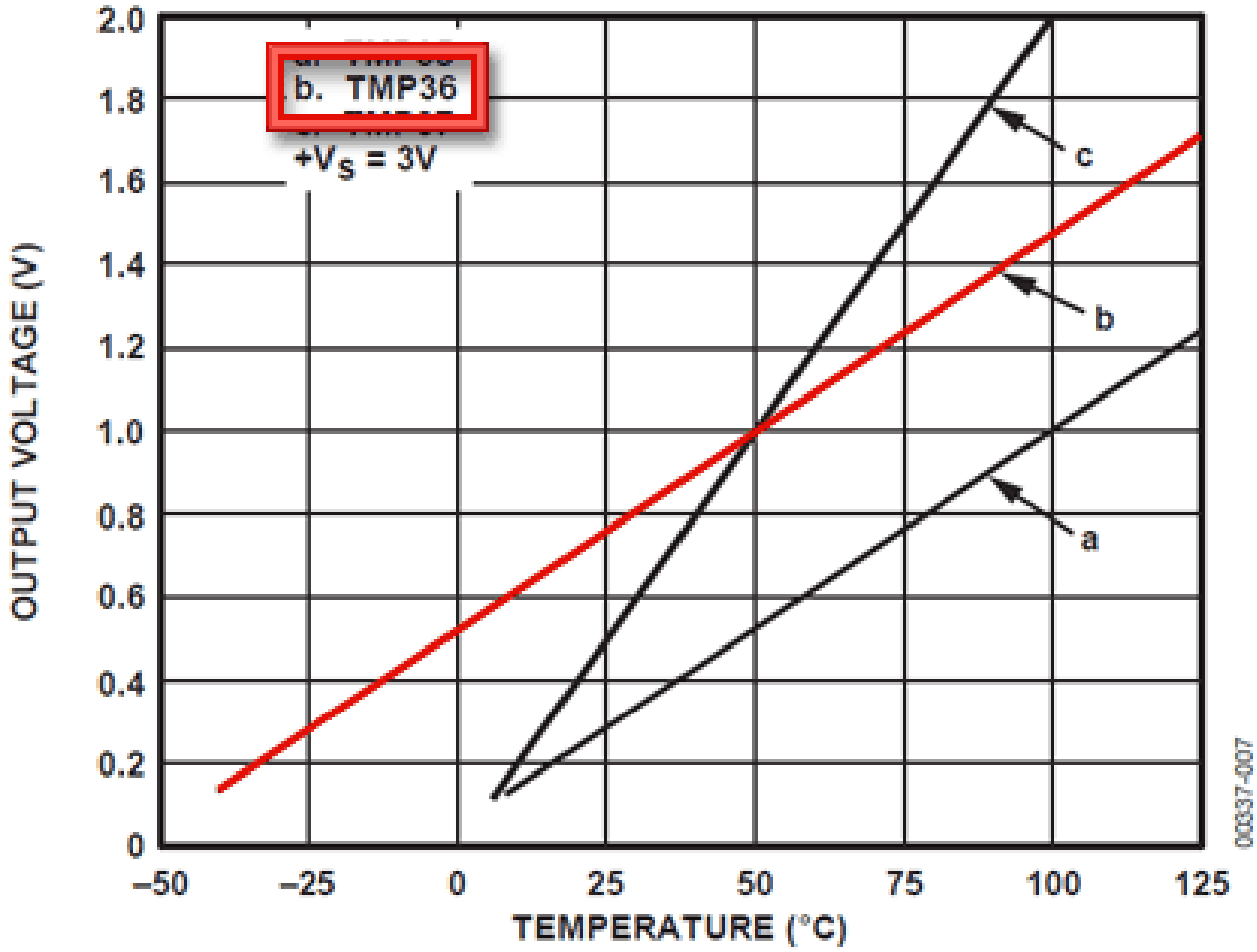
- **Tension max : 1,8V !**
- **Pas de « pin muxing » pour les entrées analogiques**
- **Fichiers virtuels**
  - `/sys/devices/platform/tsc/ain*`
    ou `/sys/devices/platform/omap/tsc/ain*`
    (dépend de la version de l'OS)
- **Plage de valeur : 1 à 4096 (12 bits)**

Capteur de température analogique

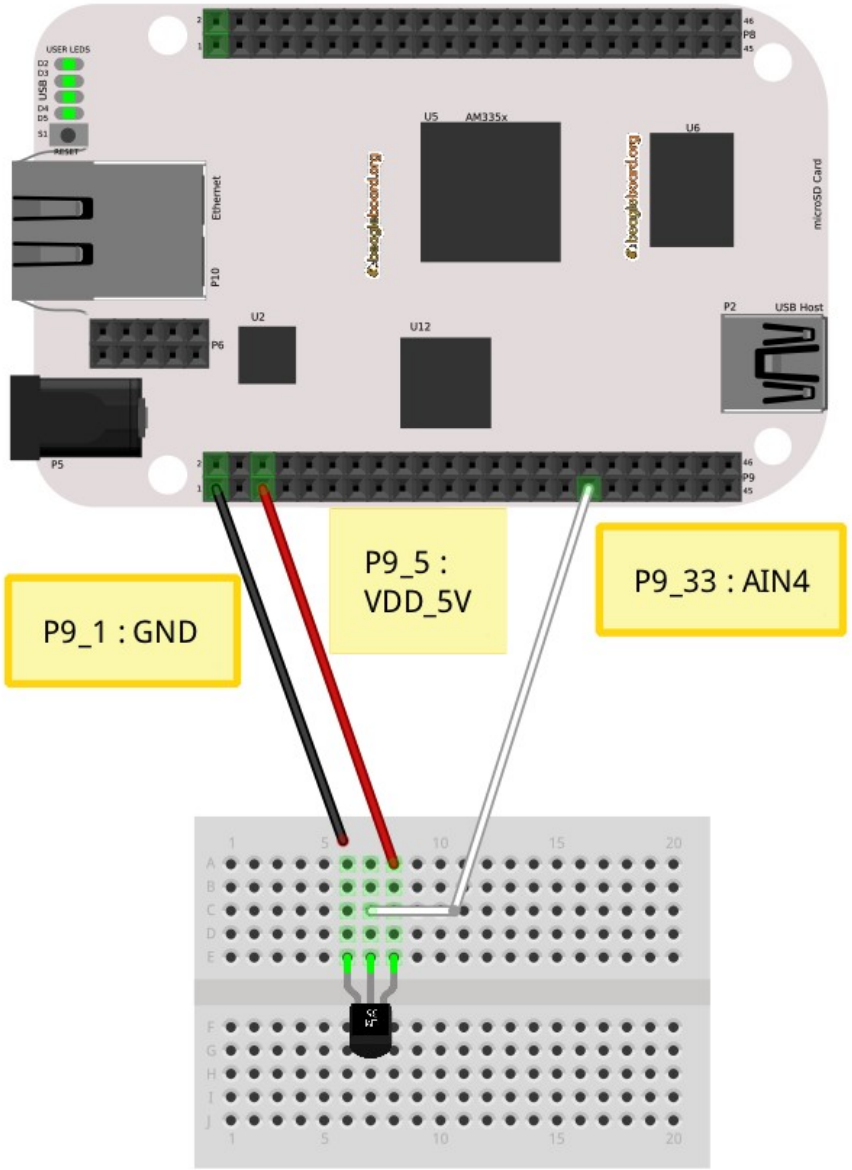
- **Capteur TMP36**

$$T\ °C = [(Vout\ en\ mV) - 500] / 10$$





Montage



- **Cablage**
  - Vout sur P9_33 (AIN4)
  - GND / 5V sur P9_1 / P9_5

Lecture de la température

- **Fichiers ain1 à ain7 pour les pin AIN0 à AIN6**
- **AIN4 dans notre cas :**

```
# cat /sys/devices/platform/omap/tsc/ain5
1660
```

- **Tension = (1660 / 4096) * 1,8 V = 729,49 mV**
- **Température = (729,49 – 500) / 10 = 22,95 °**

Bus I2C

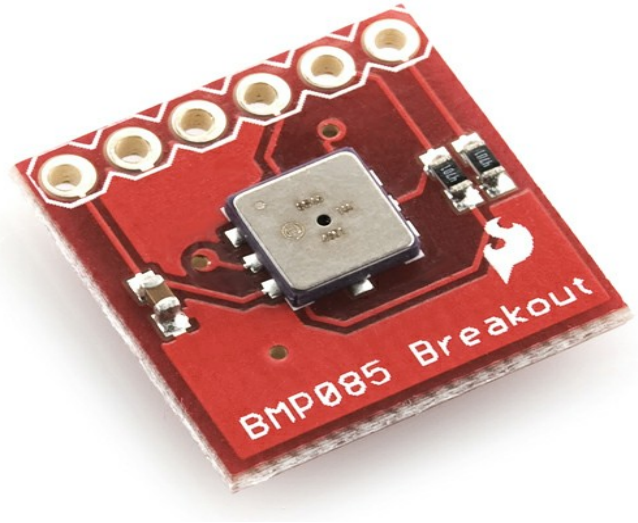
- L'anticyclone est-il loin de La Rochelle ?

Présentation I2C

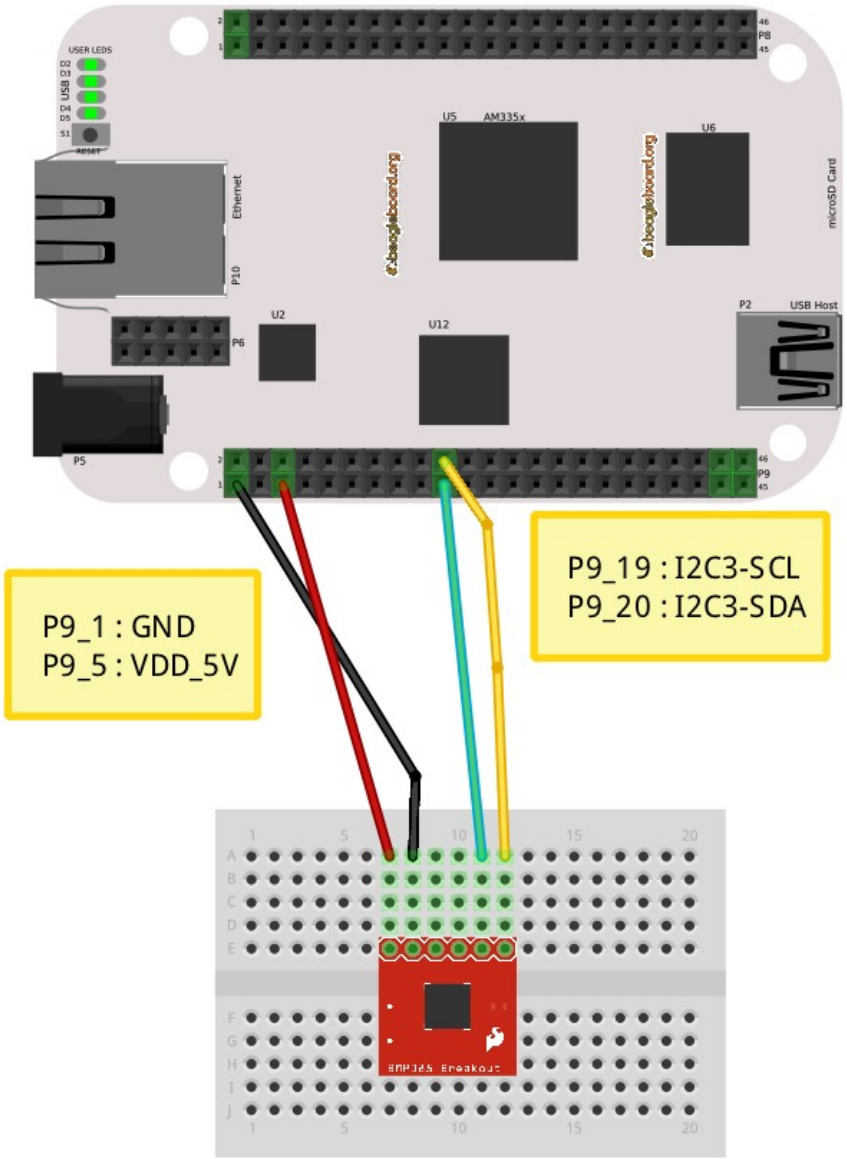
- **Standard de l'industrie des années 80 !**
  - Initié par Philips
- **Bus série à 3 fils**
  - Signal de données – SDA
  - Signal d'horloge – SCL
  - Masse
- **Plusieurs composants sur un bus**
  - Adresse pour chaque composant

Capteur de pression I2C

- **BMP085**
  - Pression atmosphérique
  - Température



Montage



Installation et détection du capteur

- **« Pin muxing » correct par défaut (P9_19 et P9_20)**
- **Détection du capteur**
  - Commande `i2cdetect`
  - Si tout est ok, l'adresse 77 est présente (celle du BMP085)

```
# i2cdetect -r -y 3
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- UU UU UU UU -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- 77
```

Utilisation

- **Prise en charge du capteur sur le bus par le noyau**

```
echo bmp085 0x77 > /sys/class/i2c-adapter/i2c-3/new_device
```

- **Lecture de la pression et de la température**

```
# cat /sys/bus/i2c/drivers/bmp085/3-0077/pressure0_input
100211
# cat /sys/bus/i2c/drivers/bmp085/3-0077/temp0_input
230
```

- **Résultat : 1002 hPa et 23 °C**

Mashup party !

Quelle JVM pour ARM ?

- **OpenJDK 6 & 7**
  - “Zero-Assembler” => Moins rapide que la VM Oracle
- **Java SE Embedded 6**
  - Conditions d'utilisation spécifiques à l'embarqué (royalties ?)
  - JRE uniquement
    - Limitations : pas de JSF, pas de Play 2, ...
- **Java SE 7 Update 6**
  - Sortie le 14/08
  - Version SFP uniquement (Soft Floating Point)
    - Version HFP (Hard Floating Point) dans la Roadmap

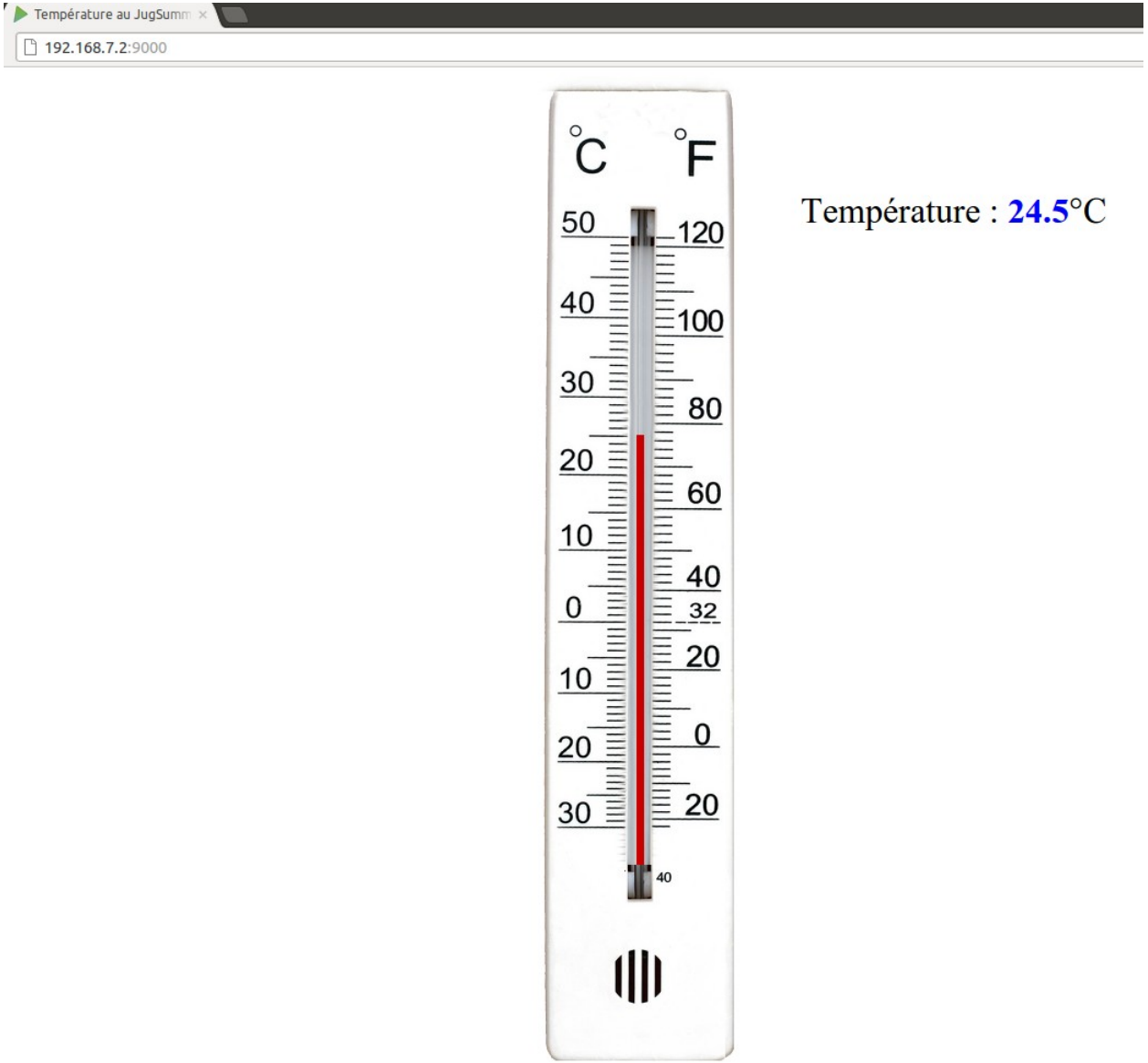
Mix de technos

- **Applatications possibles**
  - Supervision énergétique
    - Ex : Teleinfo EDF
  - Application Web “classique” multi-devices
  - Limite : Votre imagination

Démo avec play!

Une partie du code provient de : https://github.com/rheh/HTML5-canvas-projects

Plan B de la démo



Installation domotique de l'amateur Java

- **Contrôleur : Beaglebone**
  - Capteurs
  - Volets roulants
  - Monitoring des consommations
  - ...
- **IHM supervision-pilotage : Web et Android**
  - HTML5
  - Application native pour Smartphone / Tablette
- **… et une petite dose de Cloud pour les amateurs !**

Questions ?