CNAM - RSX206 - 2008

Un peu de physical computing

P. Cubaud <cubaud@cnam.fr>

1. Définitions, exemples
2. La wiimote
3. La carte Arduino
4. Couplage Arduino/Processing

1] Physical computing ?

Physical computing
From Wikipedia, the free encyclopedia
• *Have questions?*
Find out how to ask questions and get answers. •Jump to: navigation, search

Physical computing, in the broadest sense, means building interactive physical systems by the use of software and hardware that can sense and respond to the analog world. While this definition is broad enough to encompass things such as smart automotive traffic control systems or factory automation processes, it is not commonly used to describe them. In the broad sense, physical computing is a creative framework for understanding human beings' relationship to the digital world. In practical use, the term most often describes handmade art, design or DIY hobby projects that use sensors and microcontrollers to translate analog input to a software system, and/or control electro-mechanical devices such as motors, servos, lighting or other hardware.

(Greenberg, UIST'01)

INTRODUCTION

In the last decade, various movements embraced human-computer interface designs that include physical user interfaces augmented by computing power. These include *ubiquitous computing* and *calm technology* [15], *pervasive computing* [1], *tangible user interfaces* [7], *information appliances* [12] and *context-aware computing* [3].

Researchers in these areas have demonstrated many simple but exciting examples of physical user interfaces. Ishii and

Physical Computing is an approach to learning how humans communicate through computers that starts by considering how humans express themselves physically. In this course, we take the human body as a given, and attempt to design computing applications within the limits of its expression.

(Interactive Telecom. Program ITP NYU)

Les « phidgets » (S. Greenberg, C. Fitchett, U. Calgary, 2001)

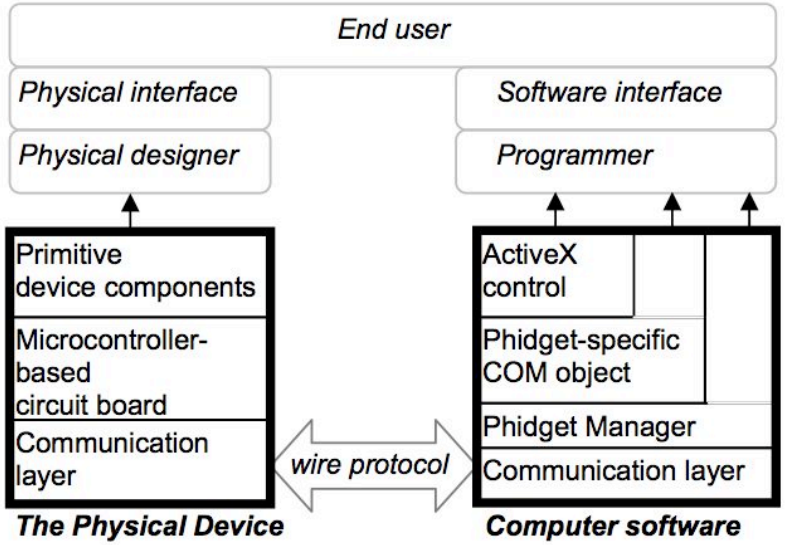
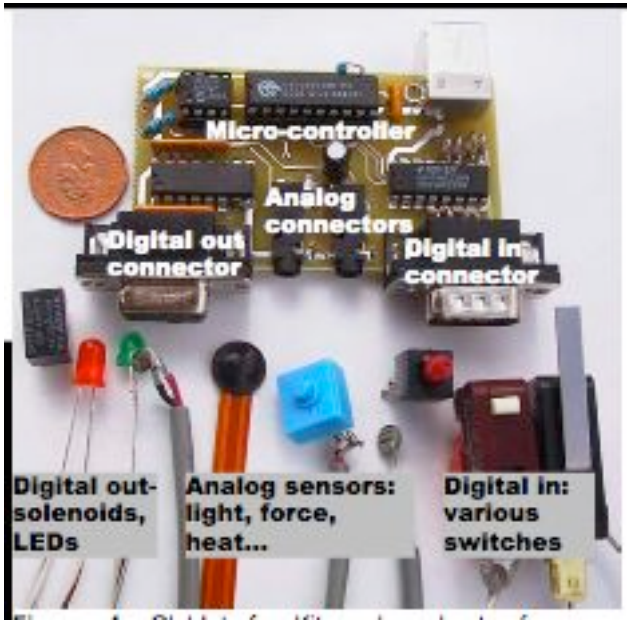
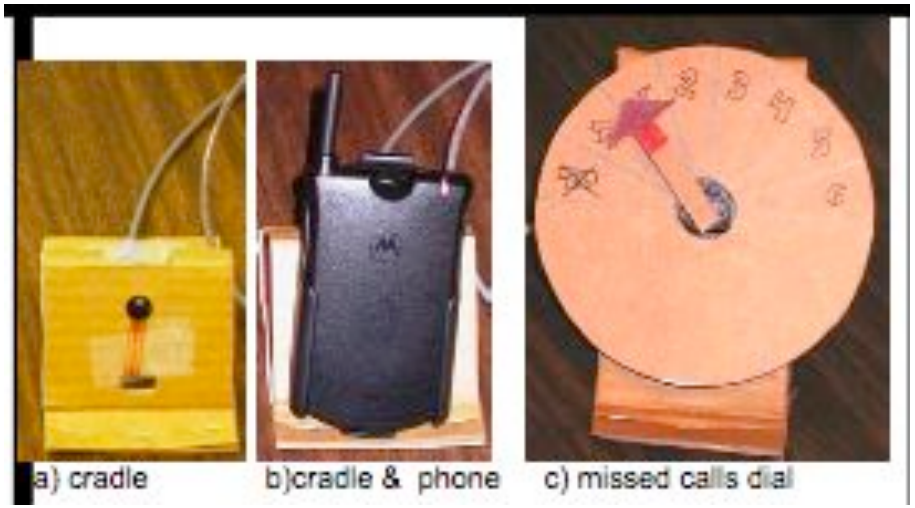


Figure 5. Phidget Architecture





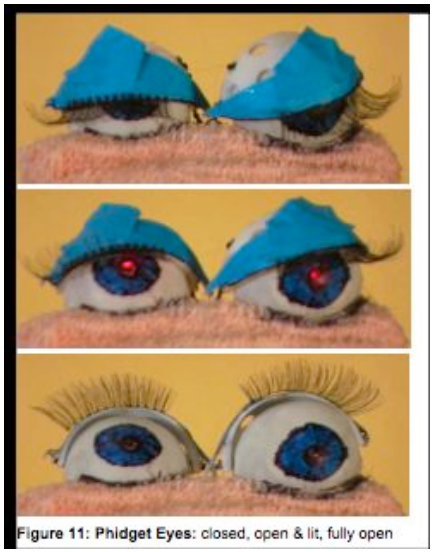
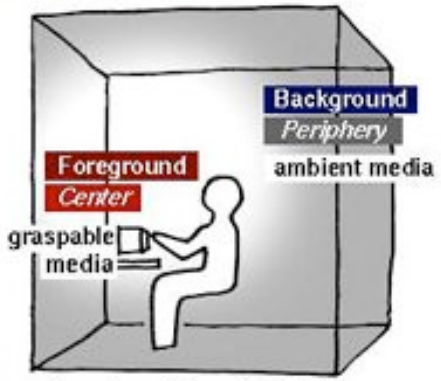


Figure 11: **Phidget Eyes:** closed, open & lit, fully open

Les interfaces tangibles

Tangible Bits is our vision of Human Computer Interaction (HCI) which guides our research in the Tangible Media Group. People have developed sophisticated skills for sensing and manipulating our physical environments. However, most of these skills are not employed by traditional GUI (Graphical User Interface). Tangible Bits seeks to build upon these skills by giving physical form to digital information, seamlessly coupling the dual worlds of bits and atoms.

Guided by the Tangible Bits vision, we are designing "tangible user interfaces" which employ physical objects, surfaces, and spaces as tangible embodiments of digital information. These include foreground interactions with graspable objects and augmented surfaces, exploiting the human senses of touch and kinesthesia. We are also exploring background information displays which use "ambient media" -- ambient light, sound, airflow, and water movement. Here, we seek to communicate digitally-mediated senses of activity and presence at the periphery of human awareness. The goal is to change the "painted bits" of GUIs (Graphical User Interfaces) to "tangible bits," taking advantage of the richness of multimodal human senses and skills developed through our lifetime of interaction with the physical world.

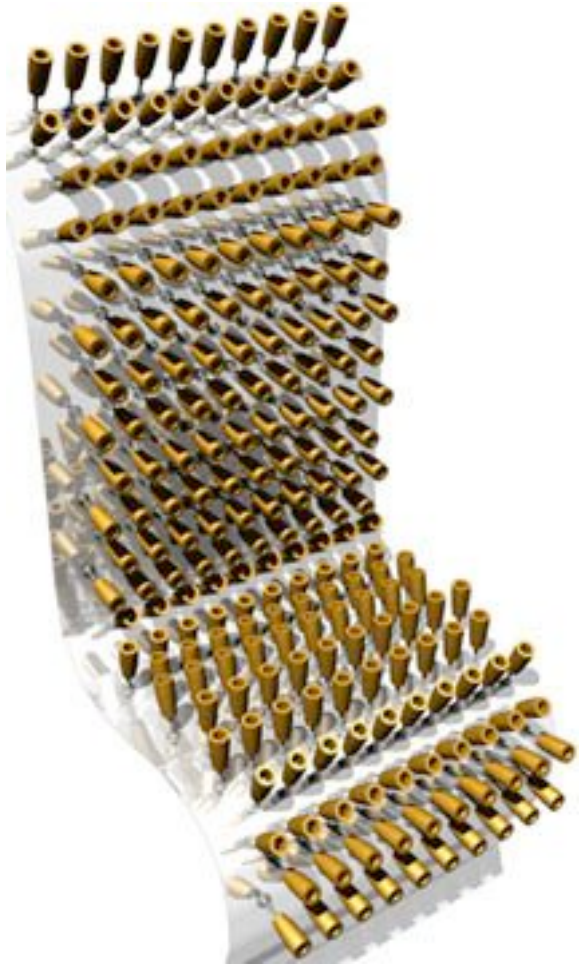


drawing: Hiroshi Ishii

Tangible Bits full paper presented at CHI 97

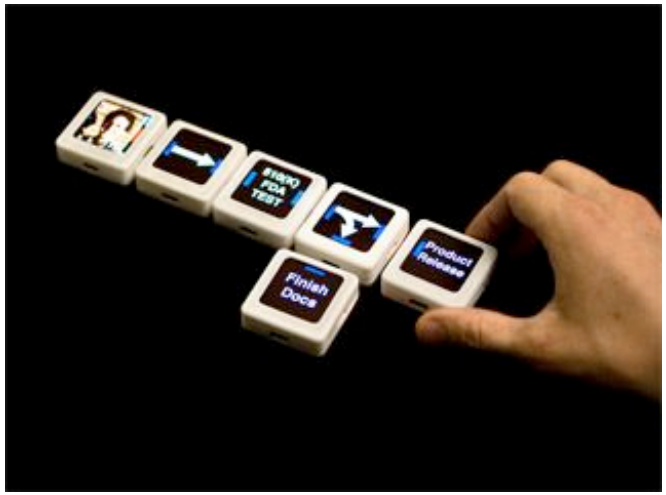
Prof. Hiroshi ISHII http://web.media.mit.edu/~ishii/

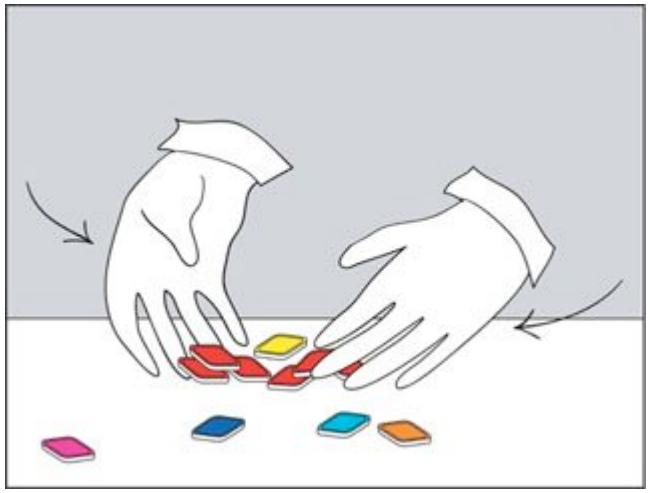
Ex. de projet de l’équipe : super cilia skin

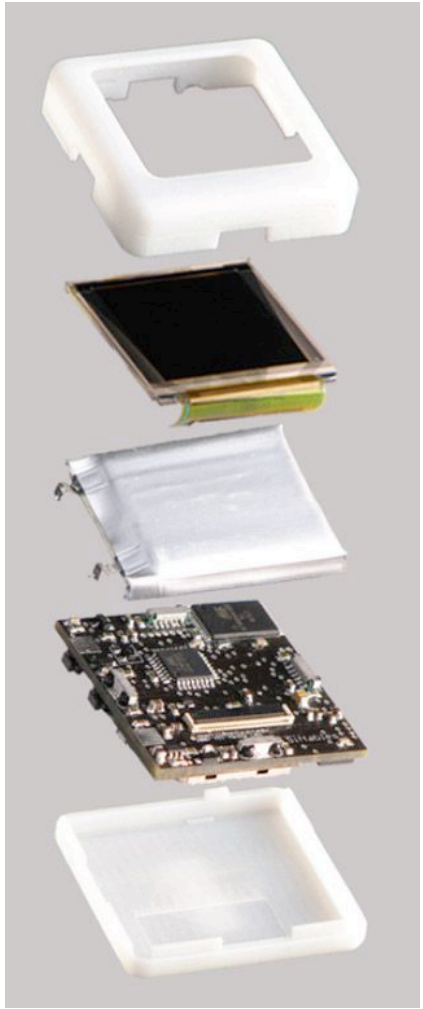


Scentsory (Nokia)

Le projet « siftables » (David Merril, MIT, 2007)

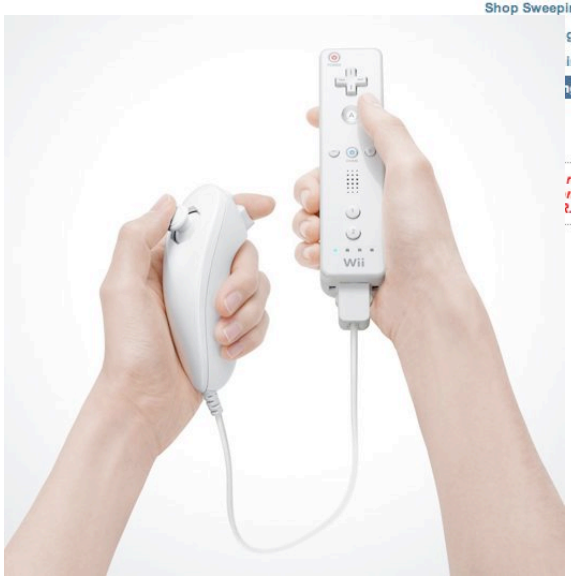


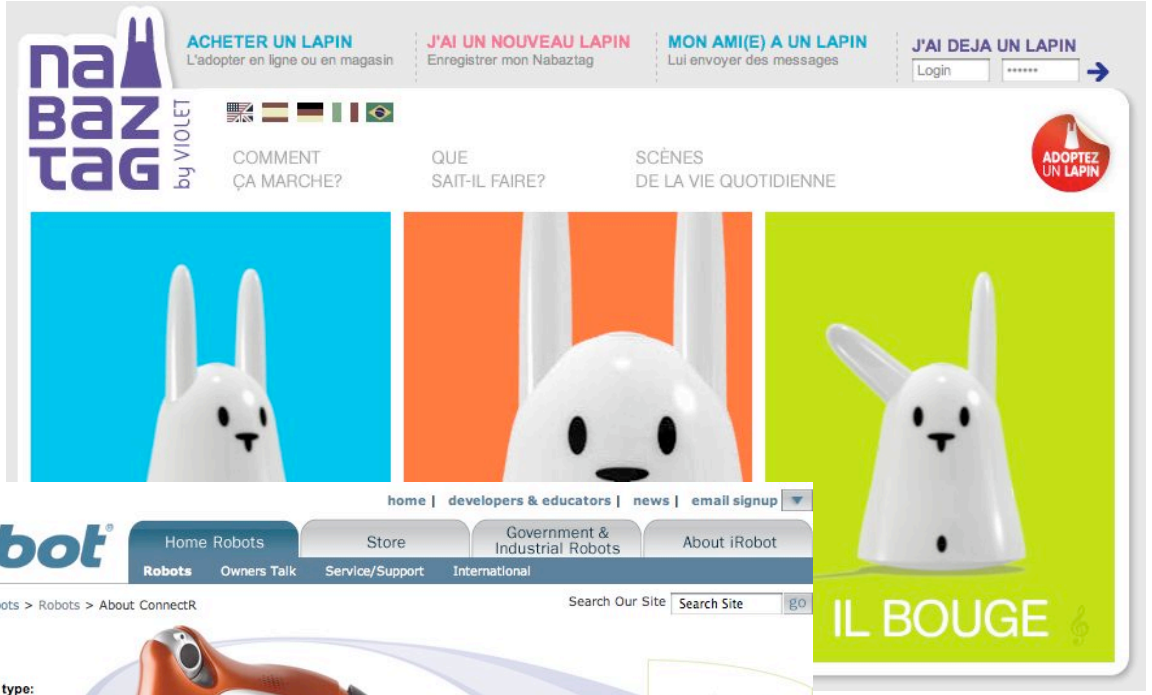


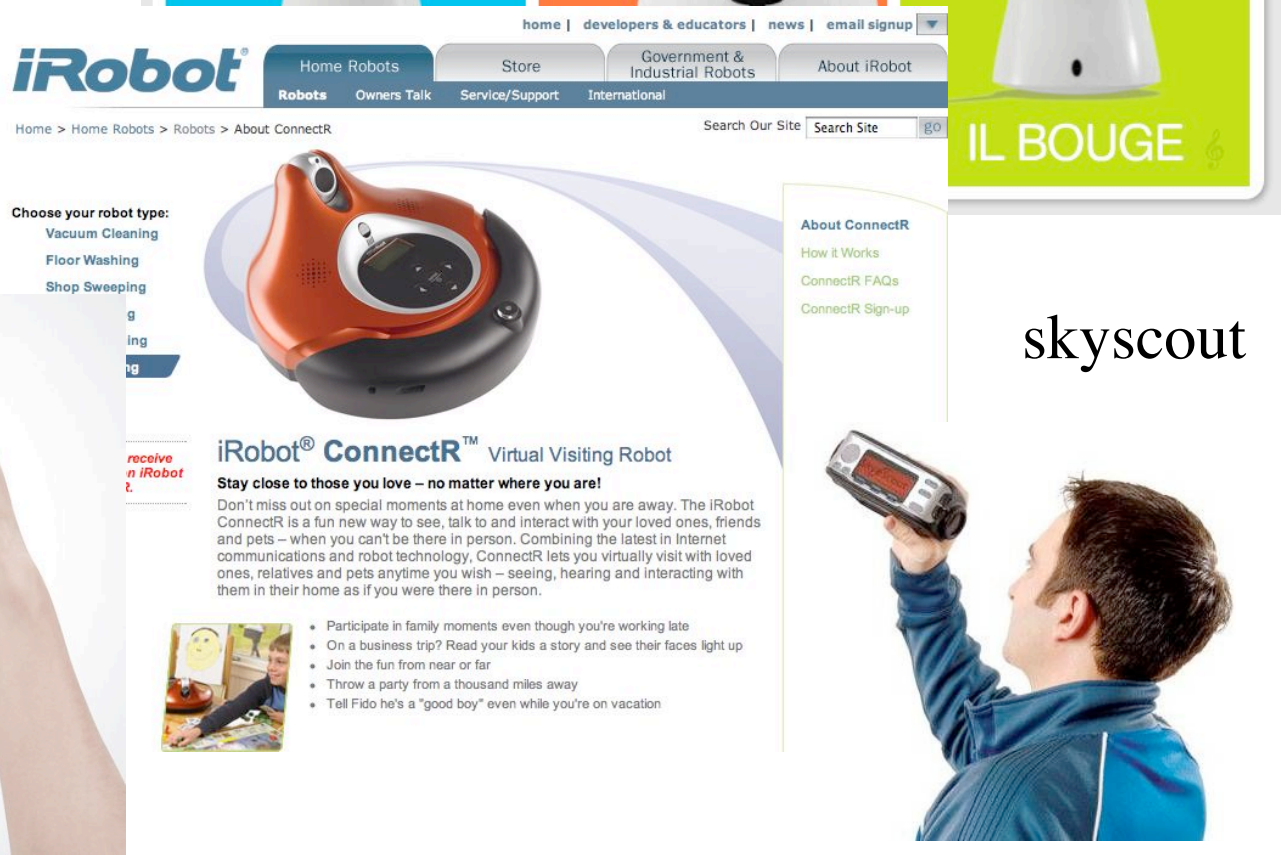


http://web.media.mit.edu/~dmerrill/siftables.html

Déjà une industrie !

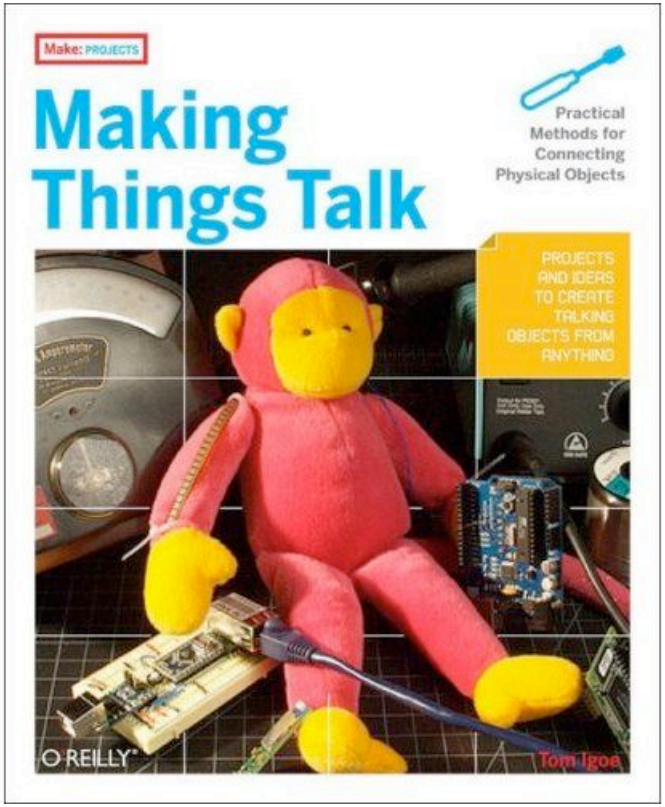


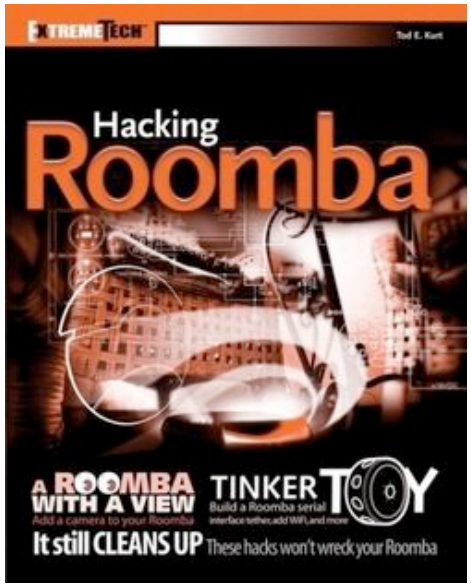


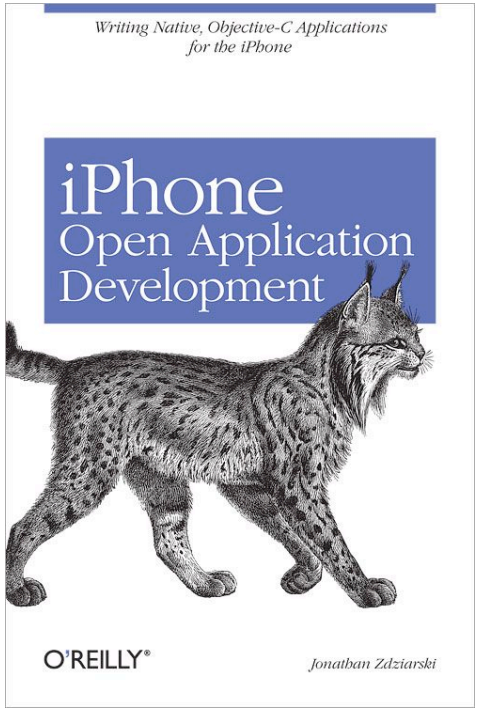


skyscout

Bibliographie

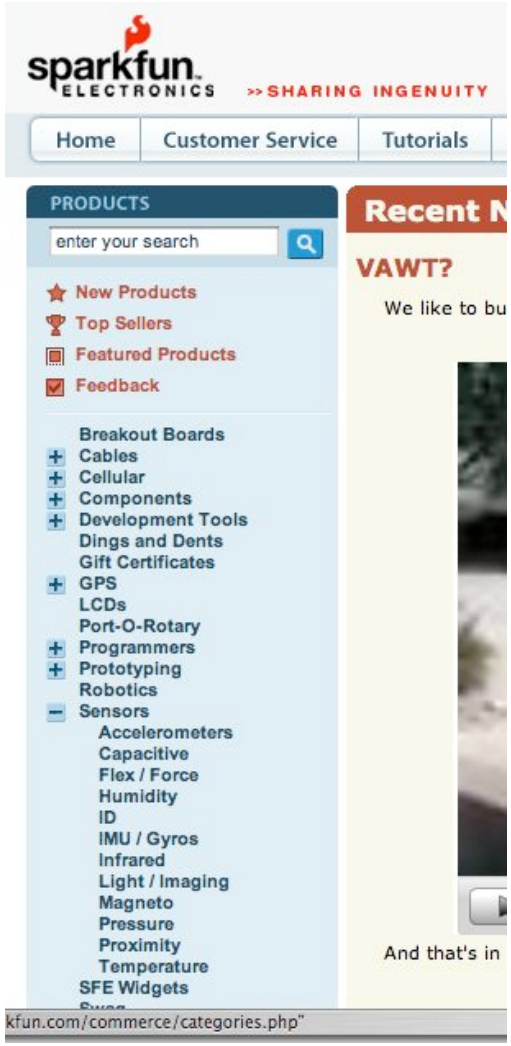






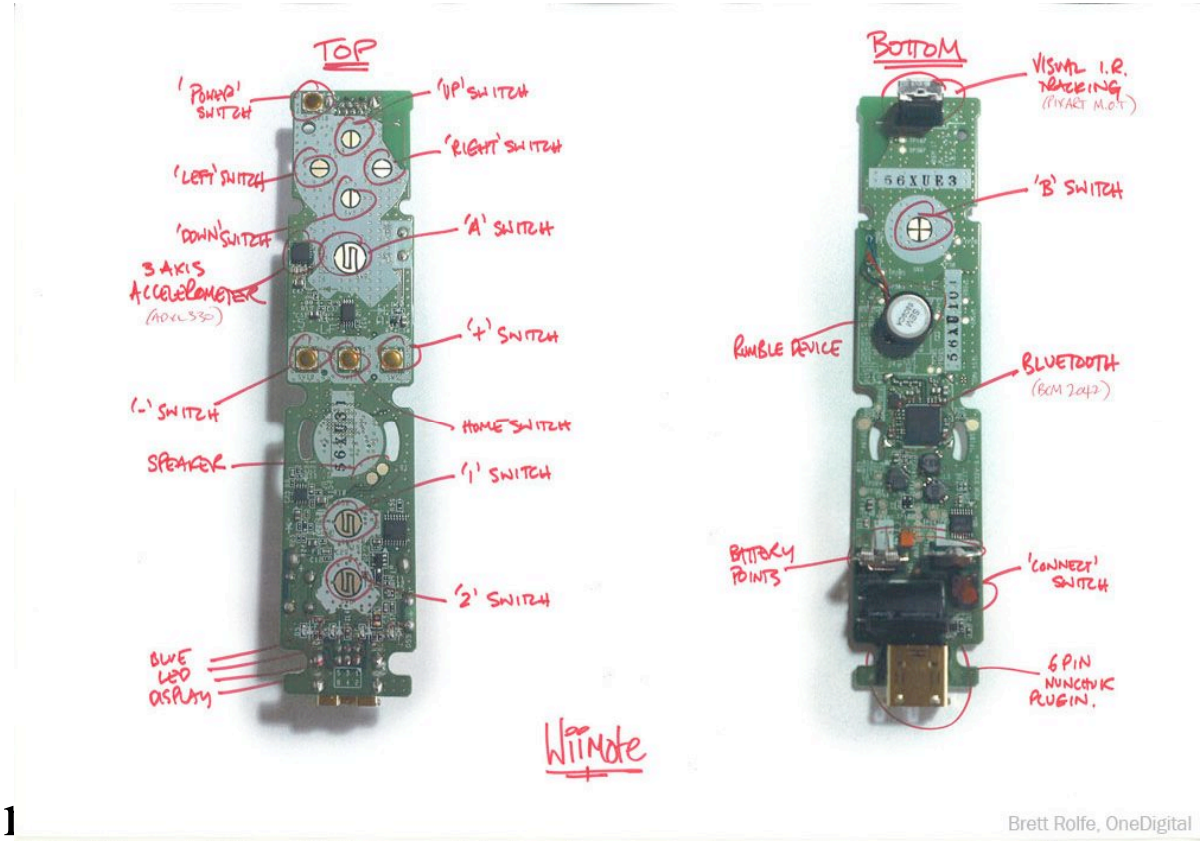
+ Designing gestural interfaces
Dan Saffer (0'Reilly ≥2008)

DIY (do it yourself) : les revendeurs



+ à Paris : StQuentin Radio, Perlor radio, etc.

2] La Wiimote de Nintendo (2006)



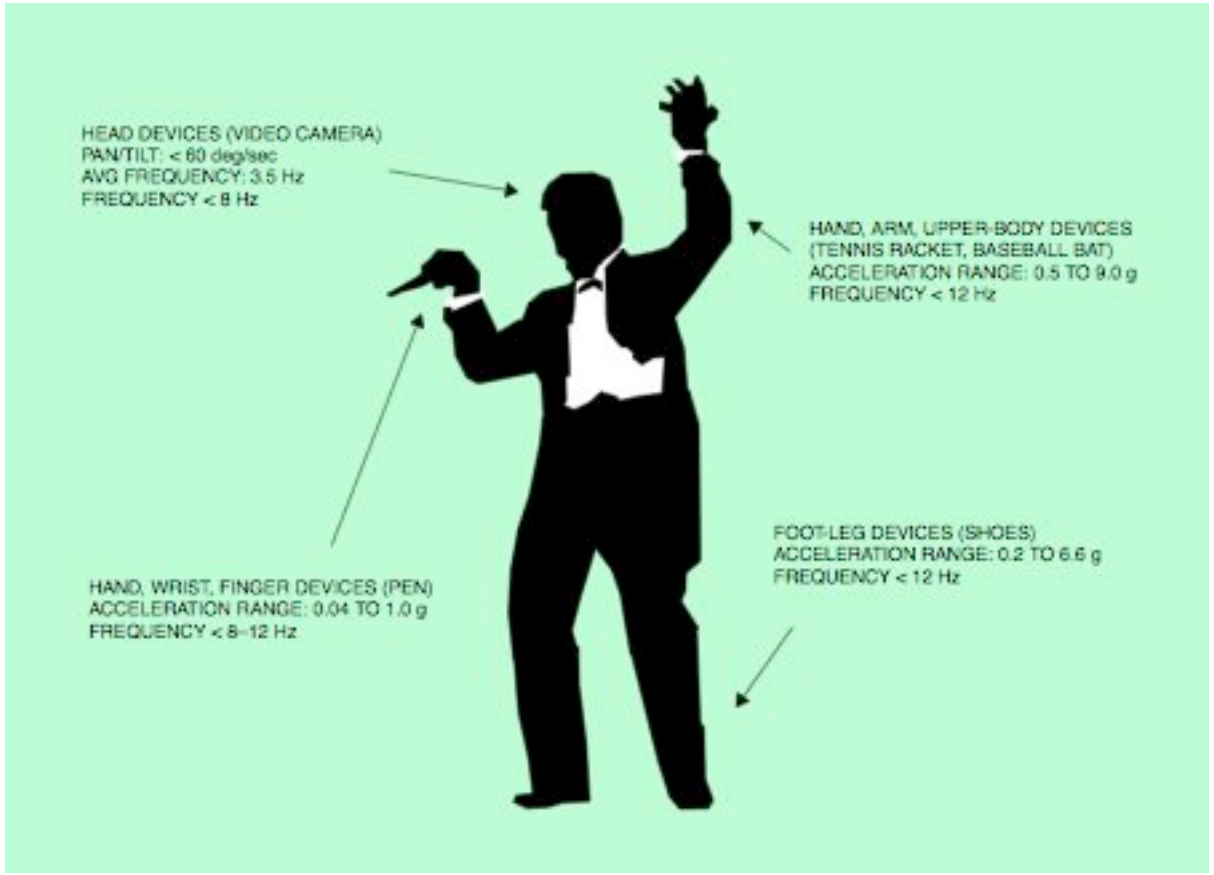
40 € wiimote + 20 €r
-accelerometre 3axes
-Camera IR + rec. Blobs
-HP, vibreur
-Plein de boutons + joysticks
-Bluetooth (et i2c avec le nunchuck)

Totalement « hacké »
=> www.wiili.com

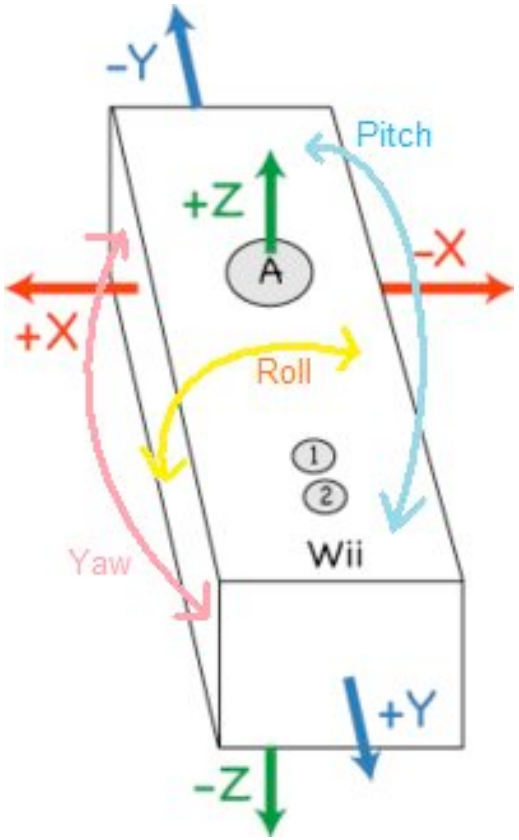
C. Verplaetse IBM Systems Journal 35(3-4) **1996** !!

Inertial proprioceptive devices: Self-motion-sensing toys and tools

by C. Verplaetse



Utilisation de l'accéléromètre



On pose la wiimote de manière à avoir successivement les trois axes X Y Z à la verticale et on collecte les valeurs renvoyées.

+Z : x1, y1, z1
+Y : x2, y2, z2
+X : x3, y3, z3

d'où les coordonnées du point origine:

x0 = (x1 +x2)/2
y0 = (y1 + y3)/2
z0 = (z2 + z3) /2

On obtient alors les coordonnées du vecteur force (exprimées en g) :

ax = (xraw -x0)/(x3-x0)
ay = (yraw - y0)/(y2-y0)
az = (zraw - z0)/(z1-z0)

http://www.wiili.org/index.php/Motion_analysis

Pour le nunchuk, il est plus difficile de faire cette calibration du fait du facteur forme. L'examen de photos d'amateurs sur le Web permet d'éviter un démontage et suggère que l'accéléromètre est placé sur un plan horizontal lorsque le nunchuck est pris en main comme un pistolet.

Calcul d'orientation pitch et roll :

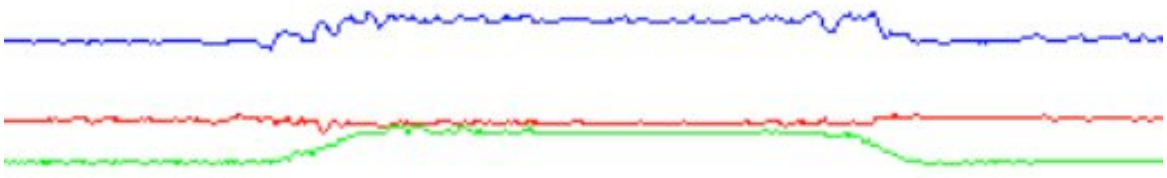
Si la wiimote n'est pas en mouvement accéléré, la mesure des cordonnées du vecteur gravité permet de d'obtenir l'orientation dans l'espace du dispositif en pitch et roll :

pitch = arctan(ax / sqrt(ay*ay + az*az))
roll = arctan(ay / sqrt(ax*ax + az*az))

source : Kionix. *Tilt-sensing with Kionix MEMS Accelerometers*. Application note AN005. 2005 (en ligne sur www.kionix.com)

On peut vérifier le repos en s'assurant que la norme ax*ax + ay*ay + az*az est proche de l'unité.

Le filtrage : de la moyenne mobile au filtre de Kalman



Dans le cas où il s'agit d'estimer une seule valeur constante subissant un bruit blanc constant de mesure (moyenne nulle, écart type R), l'itération du filtre de Kalman s'écrit en notant :

X(n) le processus étudié (dans notre exemple, l'angle de pitch ou de roll)
Z(n) la n-ième mesure : Z(n) = X(n) + V avec V suivant une loi normale N(0,R)
x(n) la n-ième valeur de l'estimateur du processus étudié

```
K := 1/(1+R) ; x := 0
repéter
        acquerir la mesure Z
        x := (1-K)*x + K*Z
        K := K/(K+1)
        utiliser l'estimateur x
jusqu'a fin des mesures
```

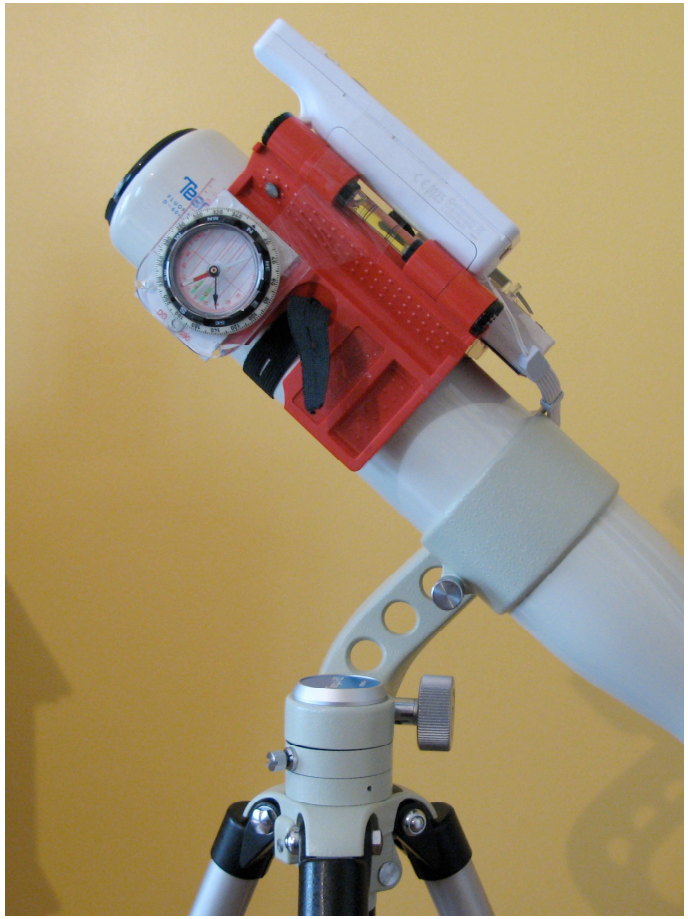
On remarque que le cas R=0 correspond pour x à un calcul de moyenne des Z(i) i=1..n

Si le processus n'est pas constant et évolue de manière aléatoire : X(n) = X(n-1) + W avec W suivant une loi normale N(0,Q), on a :

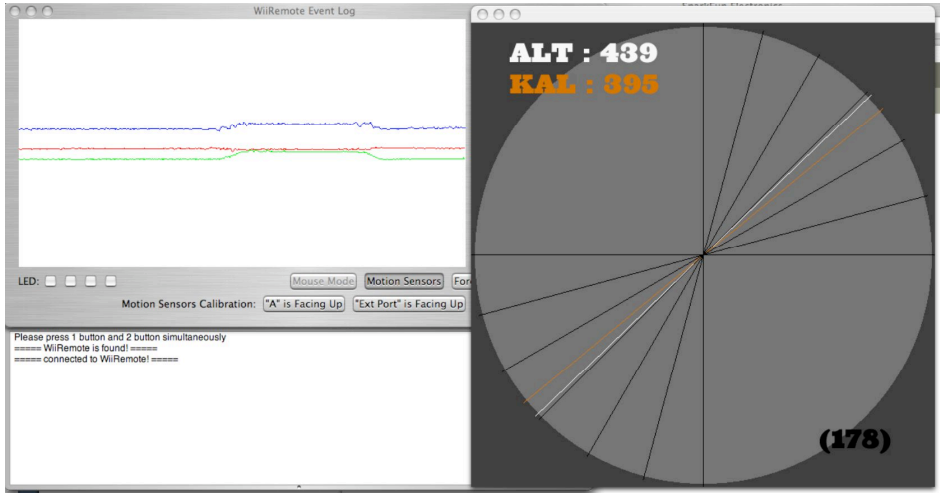
```
P := 1; x := 0;
répéter
        acquerir la mesure Z
        K = (P+Q)/(P+Q+R);
        x = (1-K)*x + K*Z;
        P = (1-K)*(P+Q);
        utiliser l'estimateur x
jusqu'a fin des mesures
```

Nous avons utilisé cette dernière procédure pour l'estimation de l'angle de pitch de la wiimote. Avec R = 0.1 et Q = 0.0001, on obtient une estimation de l'angle avec une précision d'un degré, comparable à celle d'un clinomètre de randonnée, au prix d'un temps de stabilisation assez long, de l'ordre de quelques secondes.

Un exemple d'utilisation avec Processing



Ma lunette Takahashi



Accéder à la wiimote
sur mac : JarwinRemote

(production d'un fichier txt :-( )

le code

```
void draw(){

 //saisie des infos + calculs
  wiimote_file = loadStrings("/wiimote.txt");
  if (wiimote_file.length == 1) {
   wiimote_values = wiimote_file[0].split(";");
   accX = Integer.parseInt(wiimote_values[0]);
   accY = Integer.parseInt(wiimote_values[1]);
   accZ = Integer.parseInt(wiimote_values[2]);
  }
 ax = (accX - 130.0)/26.0;
 ay = (accY - 129.5)/27.0;
 az = (accZ - 127.5)/24.0;
 //pitch = atan2(sqrt(ay*ay+az*az),ax)*180/PI;
 roll = atan2(sqrt(ax*ax+az*az),ay);
 roll += PI/180; //pour ameliorer le calibrage

 // filtrage Kalman
 K = (P+Q)/(P+Q+R);
 xhat = xhat*(1.0-K)+K*roll;
 P = (1.0-K)*(P+Q);

 background(0);
 // dessine la lunette normale
  translate(width/2,height/2);
  rotate(PI/2-roll);
  stroke(250);
  line(-width/2,0,width/2,0);
  resetMatrix();

  // dessine la lunette Kalman
  translate(width/2,height/2);
  rotate(PI/2-xhat);
  stroke(204, 102, 0);
 line(-width/2,0,width/2,0);
  resetMatrix();

 // rajoute une mire
 stroke(0);
 for (int i=0; i<=90; i = i+ 15){
    translate(width/2,height/2);
    rotate(-i*PI/180.0);
    line(-width/2,0,width/2,0);
    resetMatrix();
 }
 noStroke();
 delay(100);
}
```

Très nombreuses autres applications / détournements…

Johnny Chung LEE
HCII - CMU

Xsens

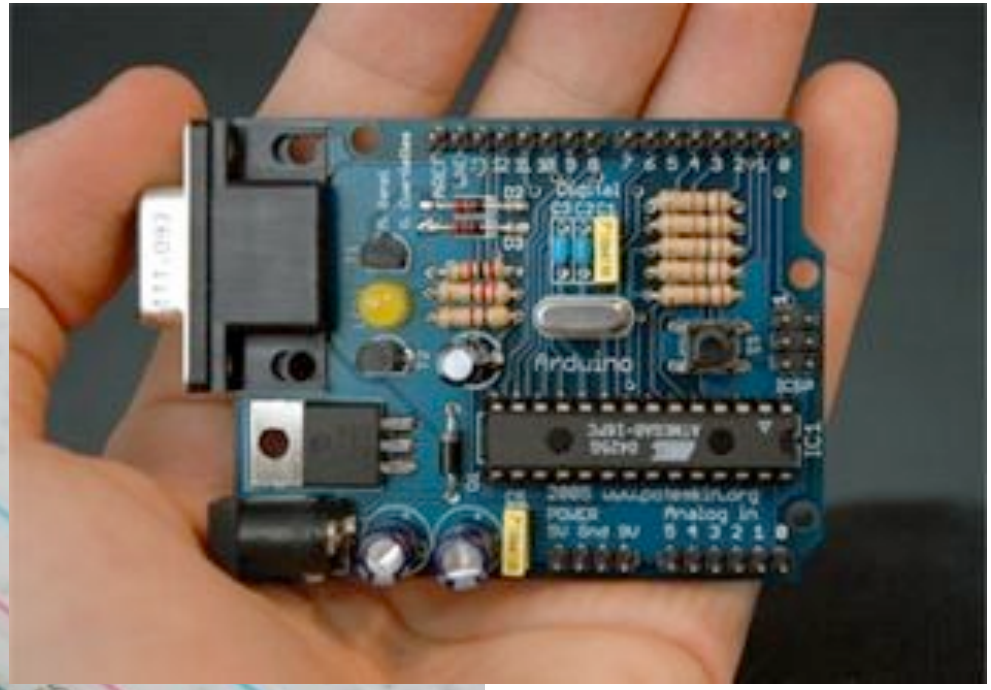
Autres produits du même genre:

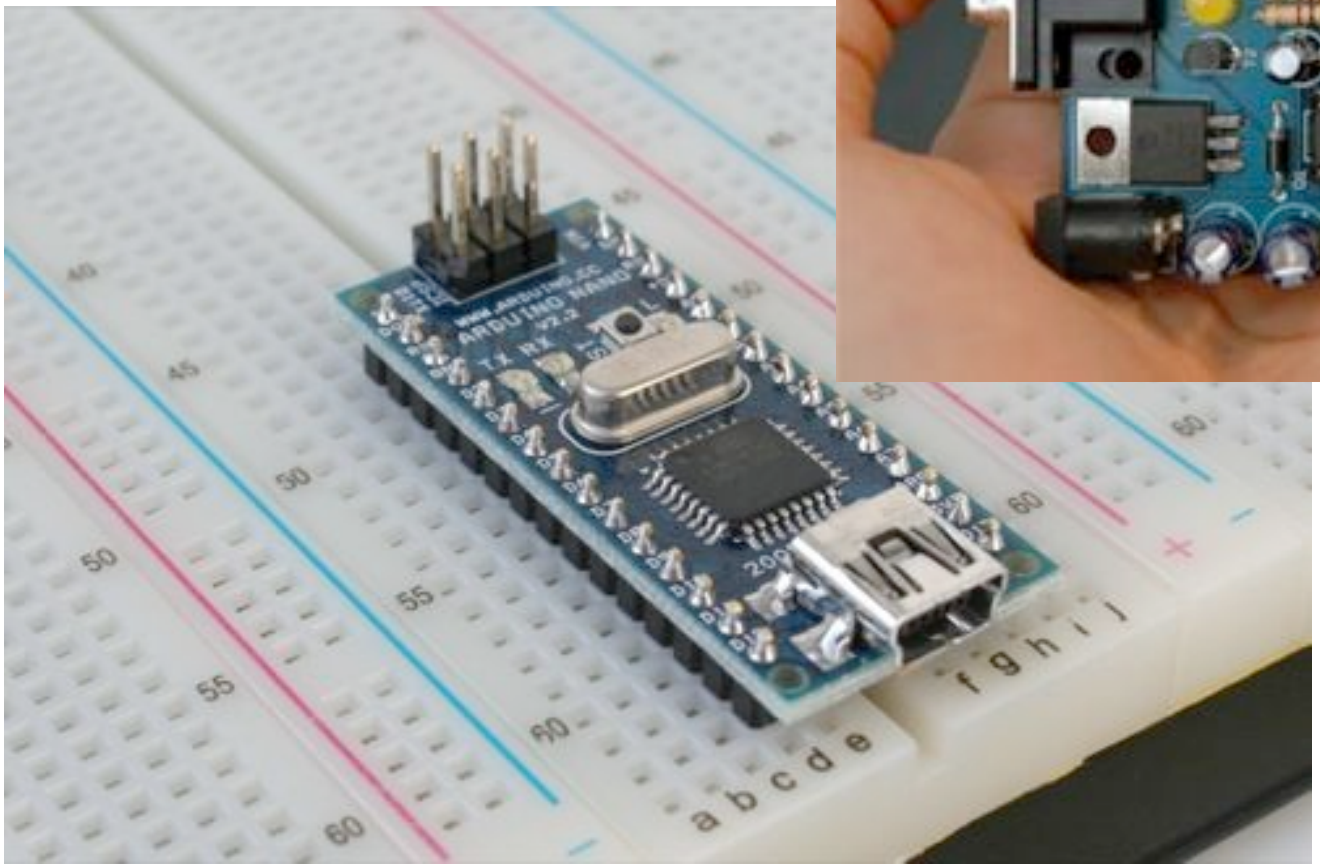
Sparkfun 450$

http://www.sparkfun.com/commerce/product_info.php?products_id=8454

3] La carte ARDUINO

www.arduino.cc





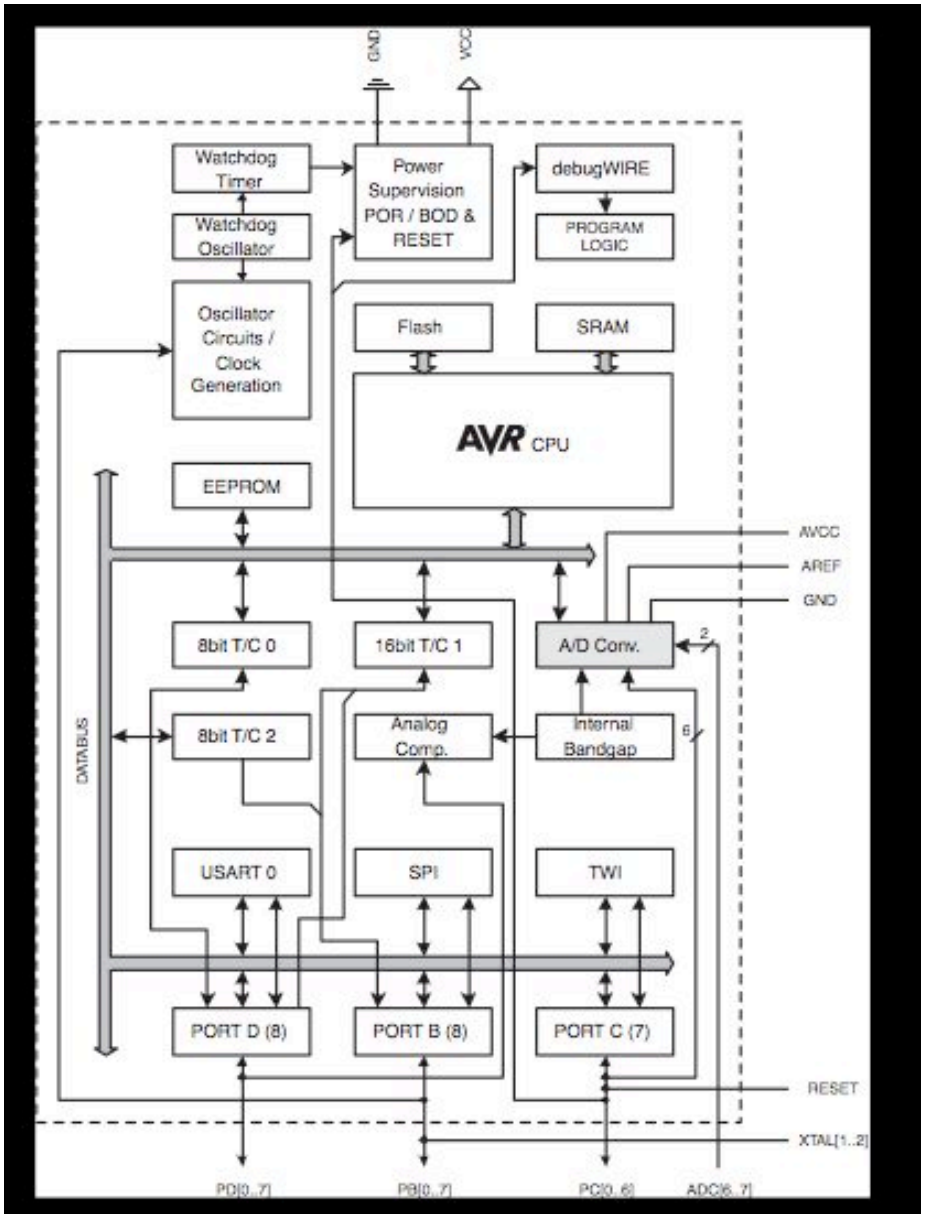
+ une version bluetooth

En France : AlyaSoft (à Bézier) : 22 euros

Le μcontroleurs ATMega
de ATMEL:

Risc 24 MIPS / 24 MHz horloge
131 instructions, 32*8 registres
16 Ko FLASH + 1 Ko SRAM
+ 512b EEPROM

La carte ARDUINO reprend
toutes ses E/S :
-6 E. analogiques 10bits
-6 E/S numériques (PWM)
-port série
+ alim
+ port usb vers l'hôte



L'environnement Arduino

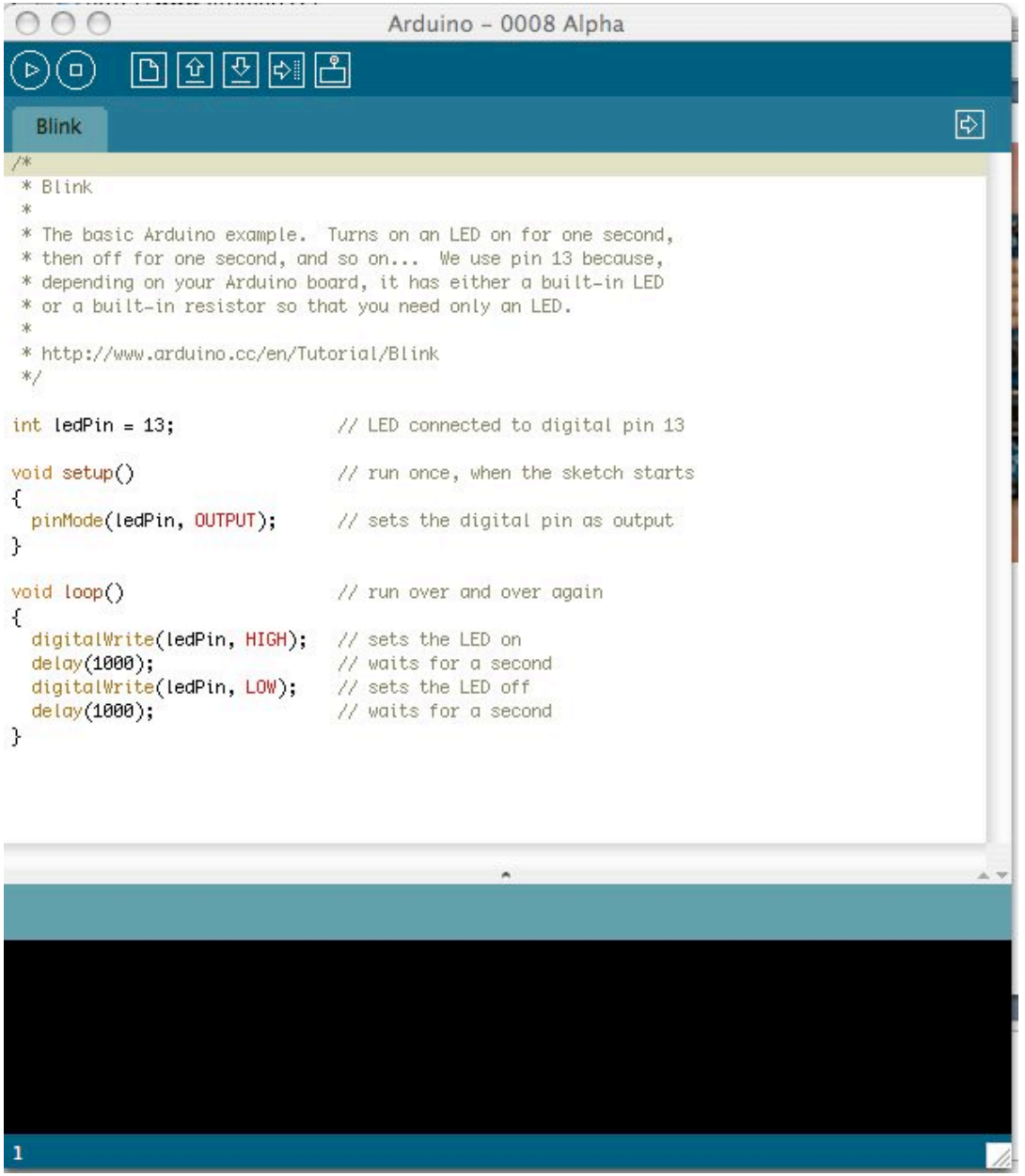
Multiplateforme !
Open source !

Basé sur C/C++

Types flottants !!
fcts math
tableaux

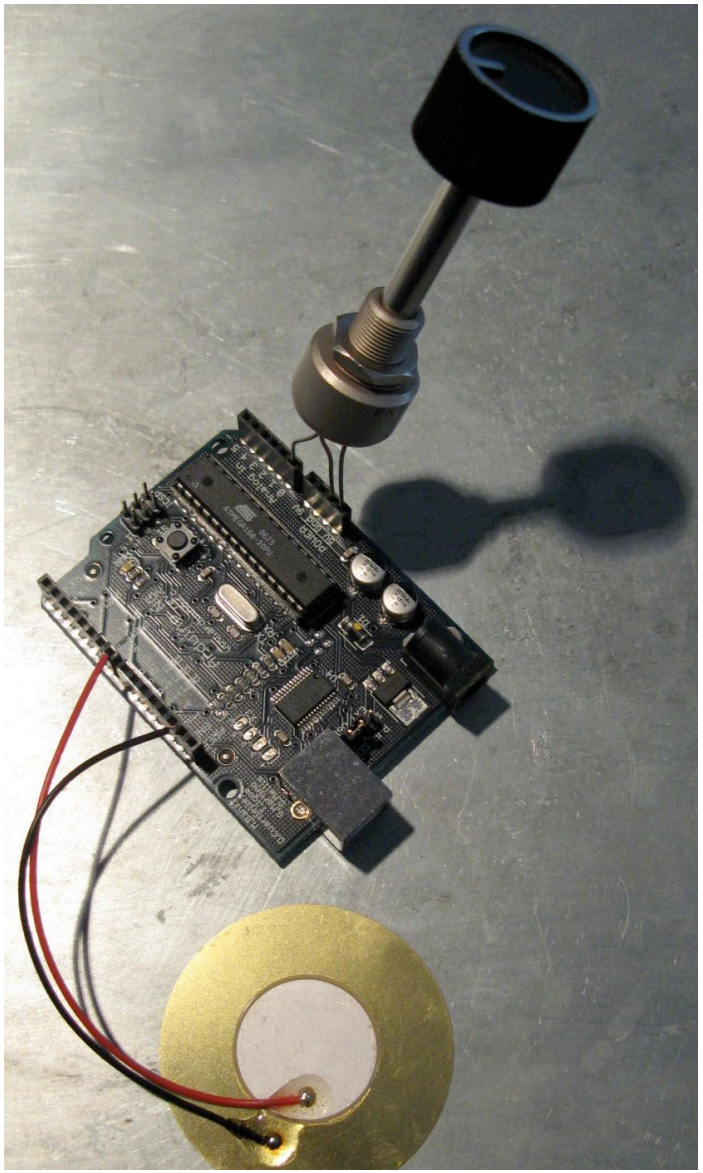
Librairies de com
Serie, i2c (Wire)

Contrôle servos …



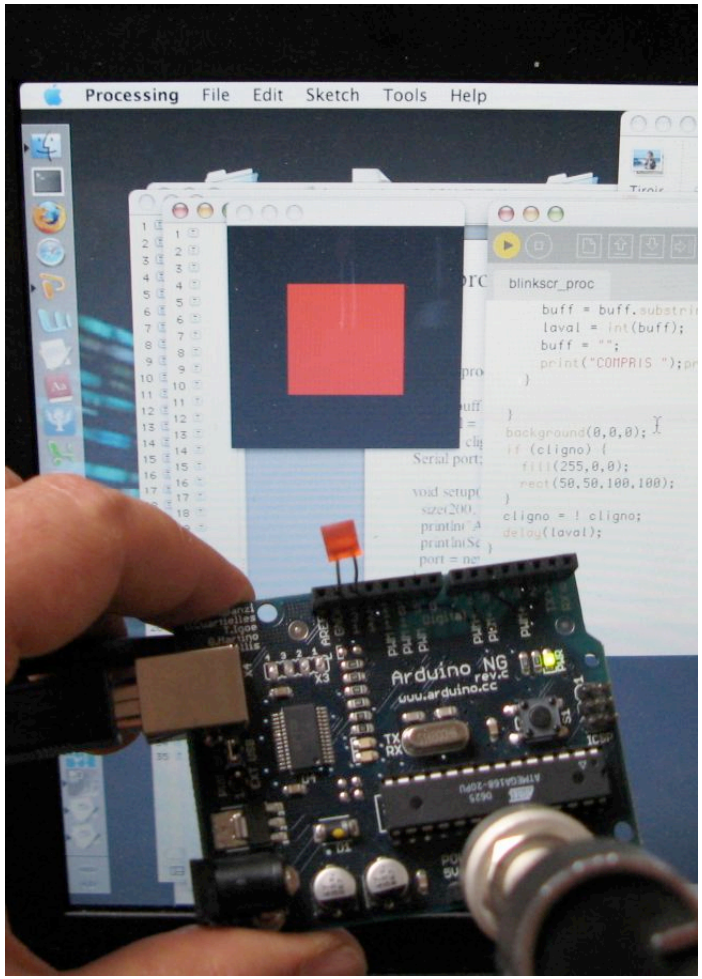
« blink » : le Hello world de l'Arduino

Buzzer :



```
/* copyleft 2006 Tod E. Kurt <tod@todbot.com
 * http://todbot.com/
 */

int potPin = 0;    // select the input pin for the potentiometer
int speakerPin = 7;

int val = 0;

void setup() {
  pinMode(speakerPin, OUTPUT);
}

void loop() {
  digitalWrite(speakerPin, LOW);
  val = analogRead(potPin);    // read value from the sensor
  val = val*2;                 // process the value a little
  for( int i=0; i<500; i++ ) {  // play it for 50 cycles
    digitalWrite(speakerPin, HIGH);
    delayMicroseconds(val);
    digitalWrite(speakerPin, LOW);
    delayMicroseconds(val);
  }
}
```

Blink, cette fois sur l'écran (avec processing)



Sur l'arduino =>

```
int potarPin = 0;
boolean ledState = LOW;
int laval;

void setup()
{
  Serial.begin(9600);
  pinMode(13, OUTPUT);
}

void loop()
{
  laval = analogRead(potarPin);
  Serial.println(laval);
  ledState = ! ledState;
  digitalWrite(13, ledState);
  delay(laval);
}
```

sur processing :

```
import processing.serial.*;

String buff = "";
int laval = 100;
boolean cligno = true;
Serial port;

void setup() {
  size(200, 200);
  println("Available serial ports:");
  println(Serial.list());
  port = new Serial(this, "/dev/tty.usbserial-A50018tg", 9600);
}
```

```
void draw() {
  if (port.available() > 0) {
   String buff = port.readStringUntil(13);
   if (buff != null) {
      print("RECU ");println(buff);
      buff = buff.substring(0, buff.length()-1);
      laval = int(buff);
      buff = "";
      print("COMPRIS ");println(laval);
    }

  }
  background(0,0,0);
  if (cligno) {
    fill(255,0,0);
    rect(50,50,100,100);
  }
  cligno = ! cligno;
  delay(laval);
}
```

Rque : le println d'Arduino envoie un CR et LF après laval

Les deux processus synchrones ???

Pour continuer :

http://todbot.com/blog/spookyarduino/