2

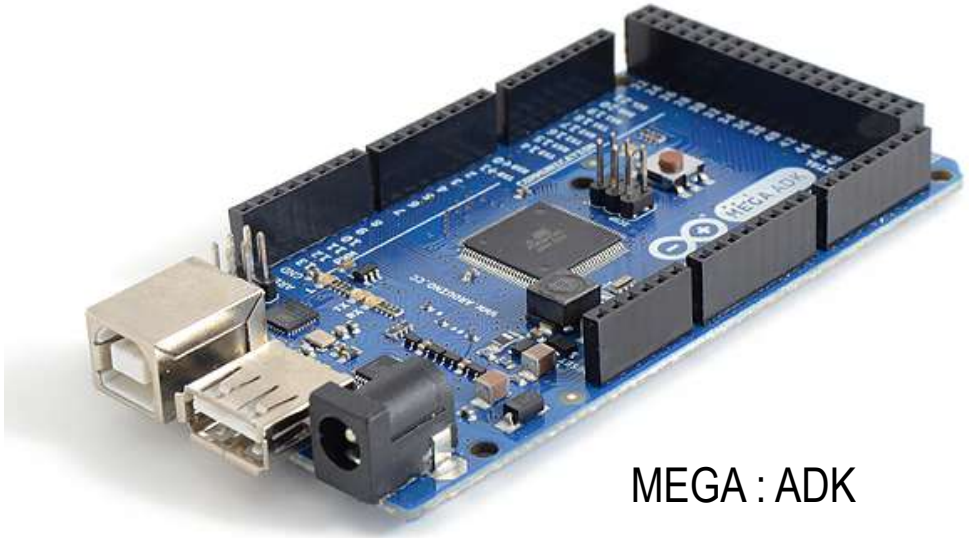
Arduino

Ahcène Bounceur

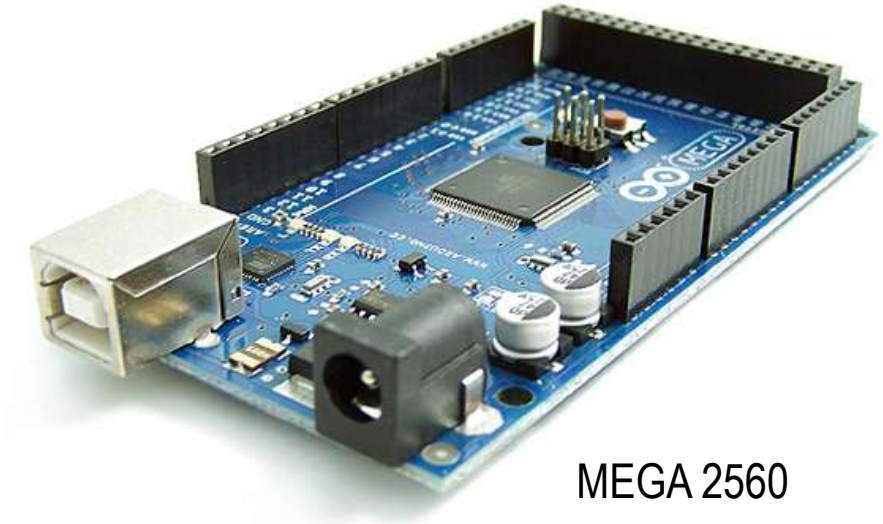
Master 2 LES
Mobilité et Objets Connectés

Cartes Arduino

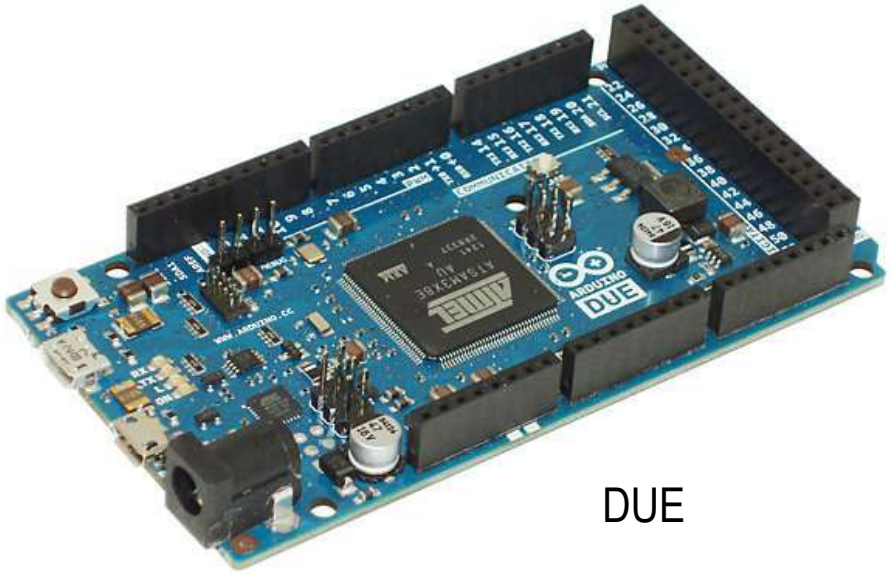
Uno
Leonardo
Duemilanove
Diecimila



MEGA : ADK



MEGA 2560



DUE

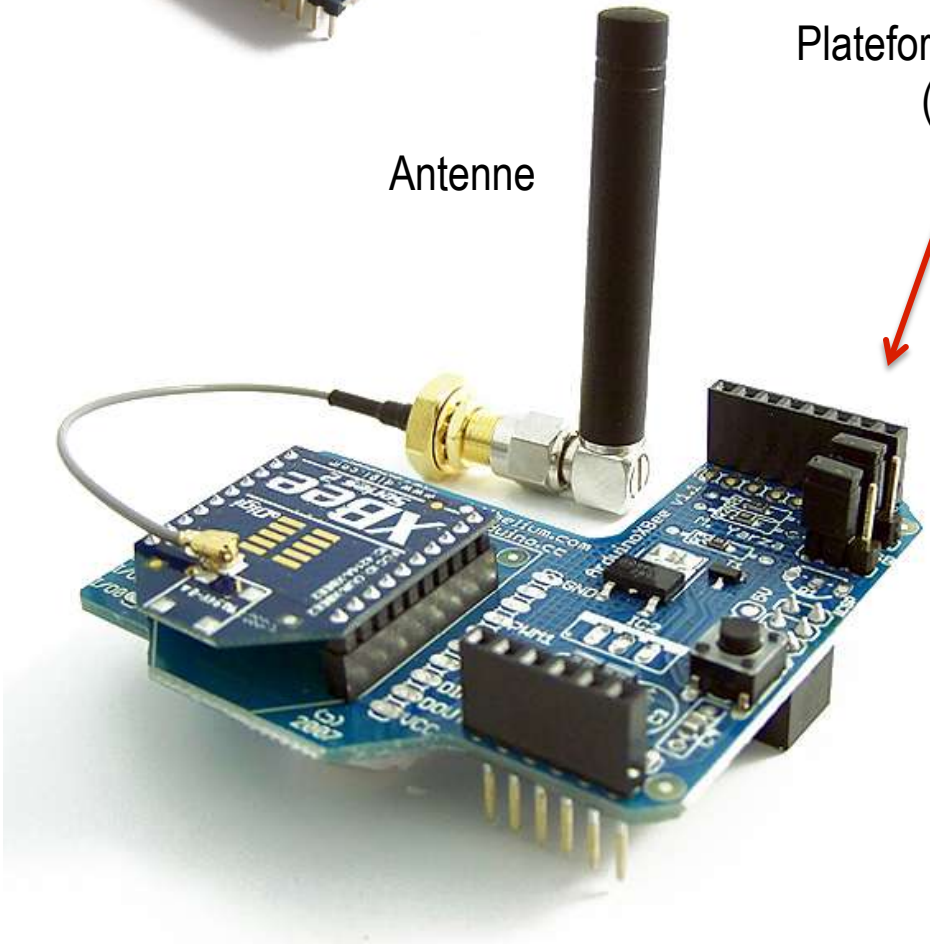
Cartes Arduino (et XBee)

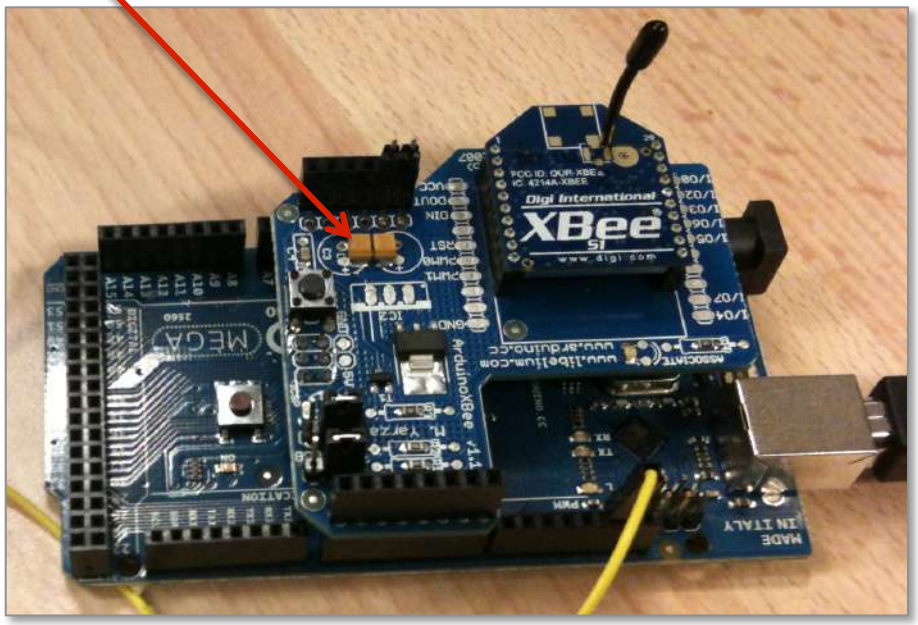
XBee

Coté Ordinateur
Adaptateur
(Gateway)

Antenne

Plateforme pour Xbee
(shield)





Cartes Arduino (Capteurs)

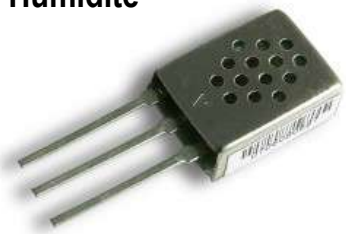
Accéléromètre

Ultrason

Mouvements

Température

Humidité



Lumière

GPS

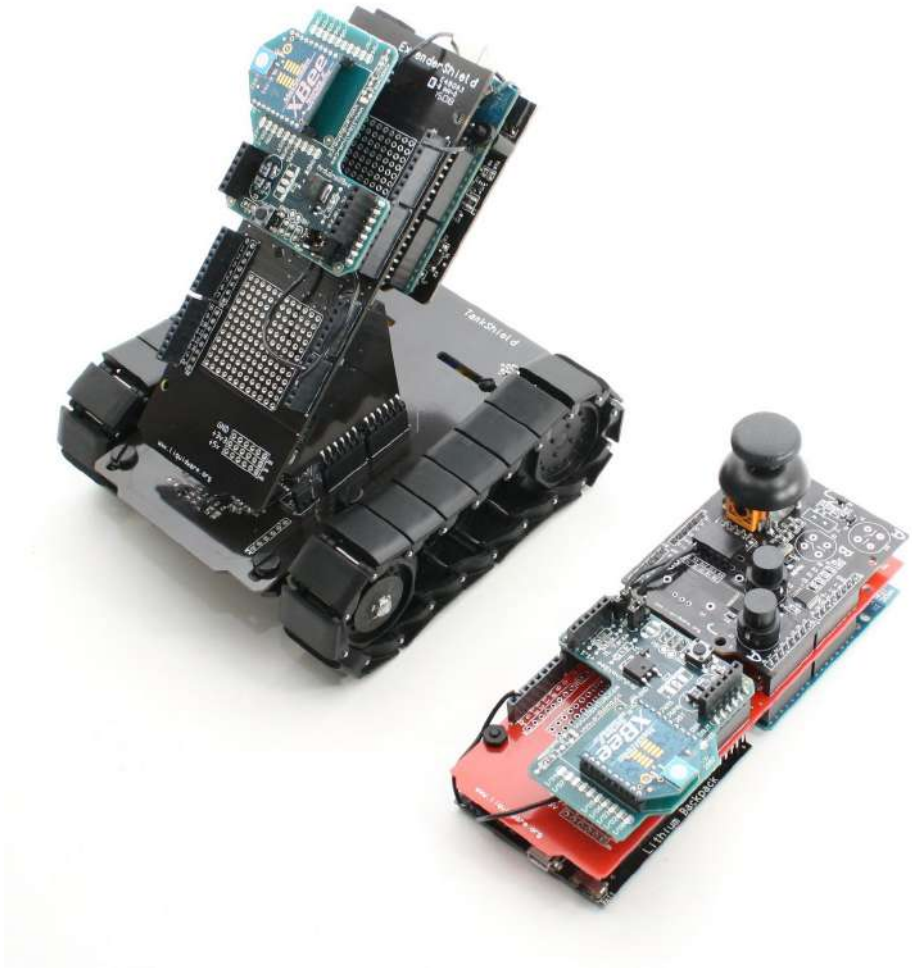
Gaz

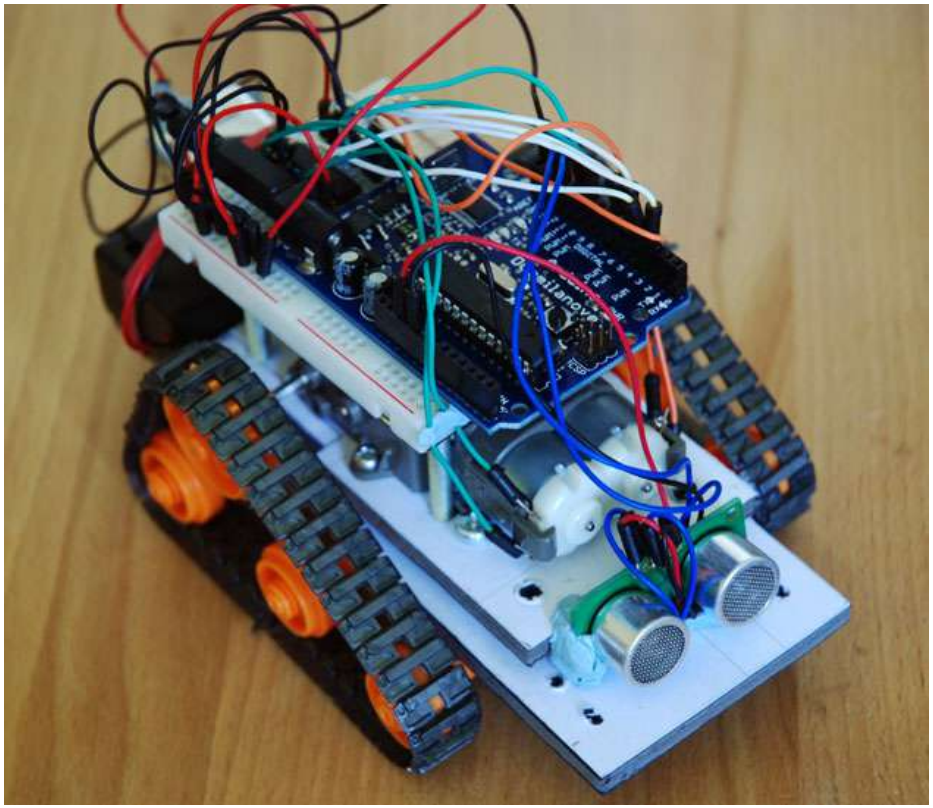
Audio

Gyroscope

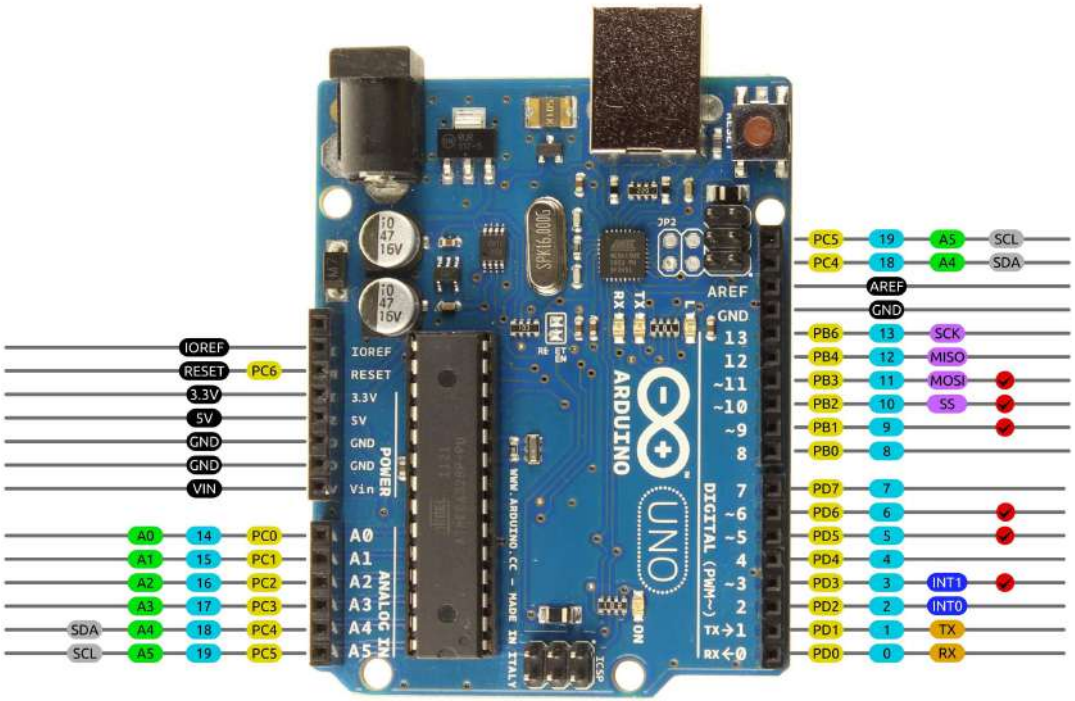
Caméra/image

Cartes Arduino (robotique)

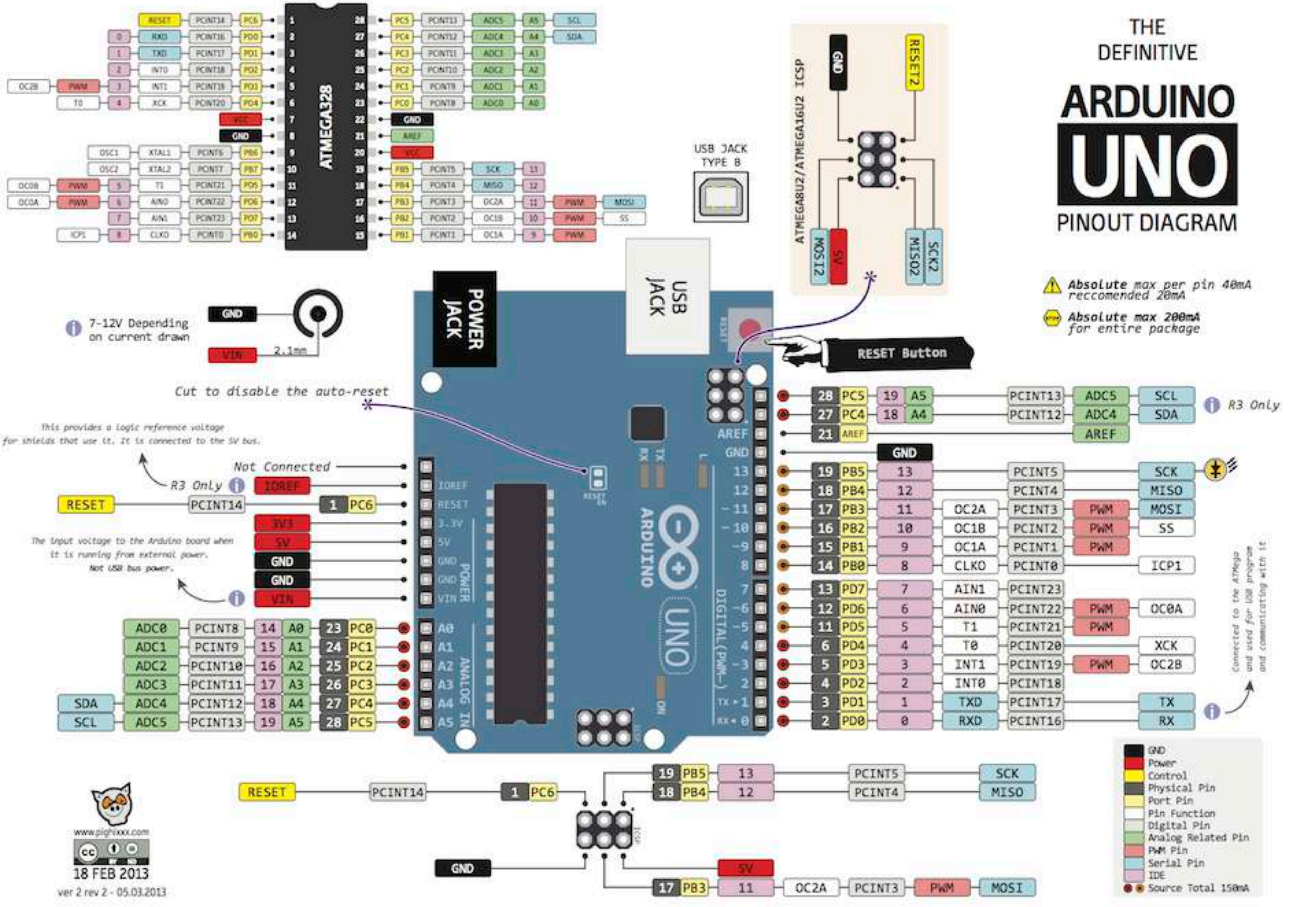




Arduino Uno R3 Pinout



AVR DIGITAL ANALOG POWER SERIAL SPI I2C PWM INTERRUPT



THE DEFINITIVE
ARDUINO
UNO
PINOUT DIAGRAM
ATMEGA328
USB JACK TYPE B
ATMEGA8U2/ATMEGA16U2 ICSP
GND
RESET2
MOSI2
5V
MISO2
SCK2
Absolute max per pin 40mA reccomended 20mA
Absolute max 200mA for entire package
POWER JACK
USB JACK
7-12V Depending on current drawn
GND
VIN
2.1mm
RESET Button
Cut to disable the auto-reset
This provides a logic reference voltage for shields that use it. It is connected to the 5V bus.
Not Connected
R3 Only
IOREF
RESET
PCINT14
1
PC6
3V3
5V
GND
GND
VIN
The input voltage to the Arduino board when it is running from external power. Not USB bus power.
ADC0 PCINT8 14 A0 23 PC0
ADC1 PCINT9 15 A1 24 PC1
ADC2 PCINT10 16 A2 25 PC2
ADC3 PCINT11 17 A3 26 PC3
SDA ADC4 PCINT12 18 A4 27 PC4
SCL ADC5 PCINT13 19 A5 28 PC5
28 PC5 19 A5 PCINT13 ADC5 SCL
27 PC4 18 A4 PCINT12 ADC4 SDA
R3 Only
21 AREF AREF
GND
19 PB5 13 PCINT5 SCK
18 PB4 12 PCINT4 MISO
17 PB3 11 OC2A PCINT3 PWM MOSI
16 PB2 10 OC1B PCINT2 PWM SS
15 PB1 9 OC1A PCINT1 PWM
14 PB0 8 CLKO PCINT0 ICP1
13 PD7 7 AIN1 PCINT23
12 PD6 6 AIN0 PCINT22 PWM OC0A
11 PD5 5 T1 PCINT21 PWM
6 PD4 4 T0 PCINT20 XCK
5 PD3 3 INT1 PCINT19 PWM OC2B
4 PD2 2 INT0 PCINT18
3 PD1 1 TXD PCINT17 TX
2 PD0 0 RXD PCINT16 RX
Connected to the ATMega and used for USB program and communicating with it
AREF
GND
DIGITAL (PWM~)
ANALOG IN
POWER
ARDUINO
UNO
RESET 1 PC6 PCINT14
19 PB5 13 PCINT5 SCK
18 PB4 12 PCINT4 MISO
GND
5V
17 PB3 11 OC2A PCINT3 PWM MOSI
ICSP
GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
PWM Pin
Serial Pin
IDE
Source Total 150mA
www.pighixxx.com
18 FEB 2013
ver 2 rev 2 - 05.03.2013

Arduino (UNO R3)

KB = Ko = Kilo octets

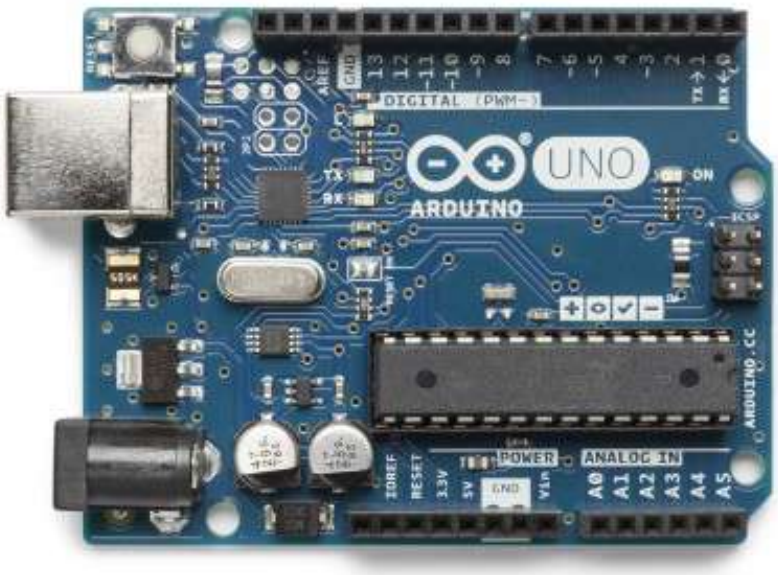
La carte Arduino UNO R3 est une carte à microcontrôleur basée sur un **ATmega328P**

ATmega328P	32Ko (Flash)	1Ko (EEPROM)	2Ko (SRAM)	15MHz (Horloge)

14 Entrées/Sorties numériques
dont 6 sorties analogiques (PWM)

2pour la communication

USB



Connecteur
d'alimentation
jack (7-12V)

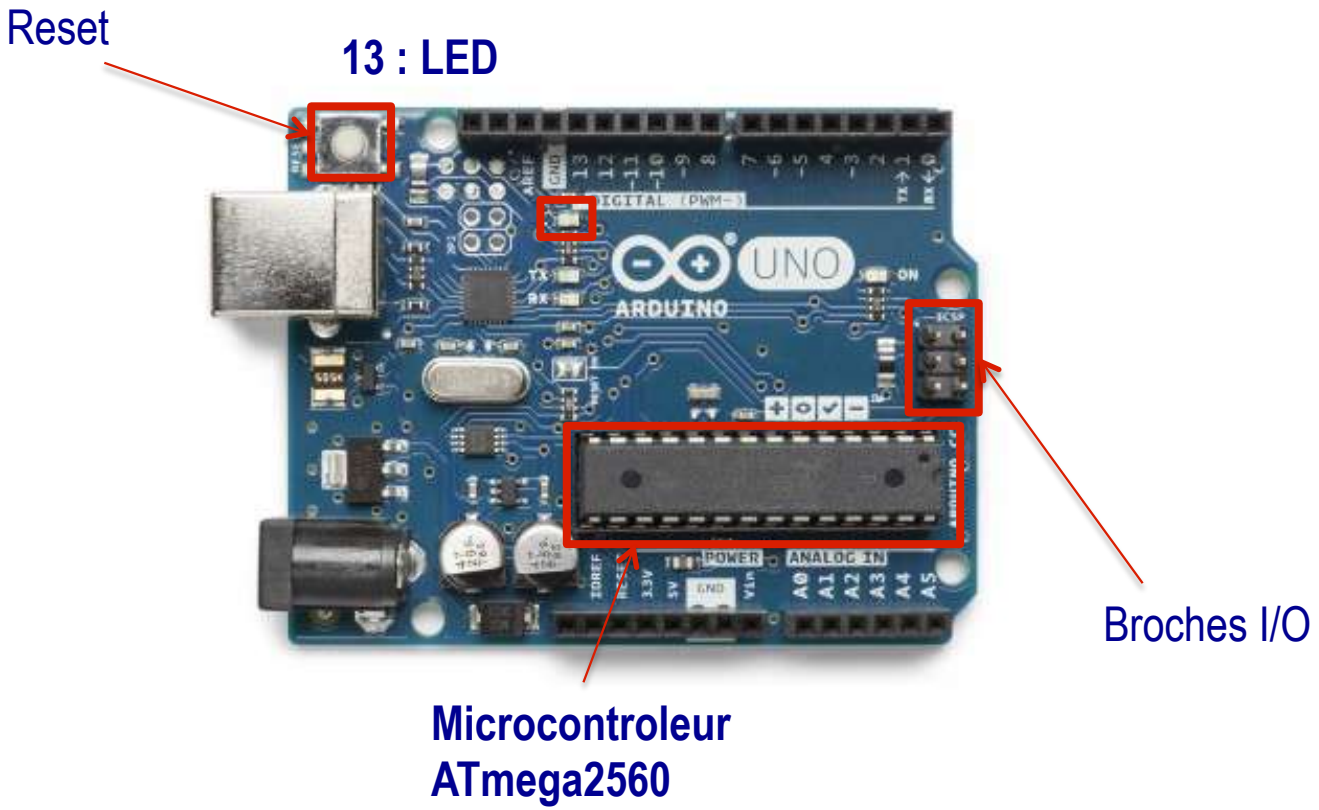
Alimentation
RESET, 3V, 5V
2 GND et 1 Vin

6 Entrées analogiques
ou 6 Entrées/Sorties numériques

Arduino (UNO R3)

La carte Arduino UNO R3 est une carte à microcontrôleur basée sur un **ATmega328P**

ATmega328P	32Ko (Flash)	1Ko (EEPROM)	2Ko (SRAM)	15MHz (Horloge)



Arduino (MEGA 2560)

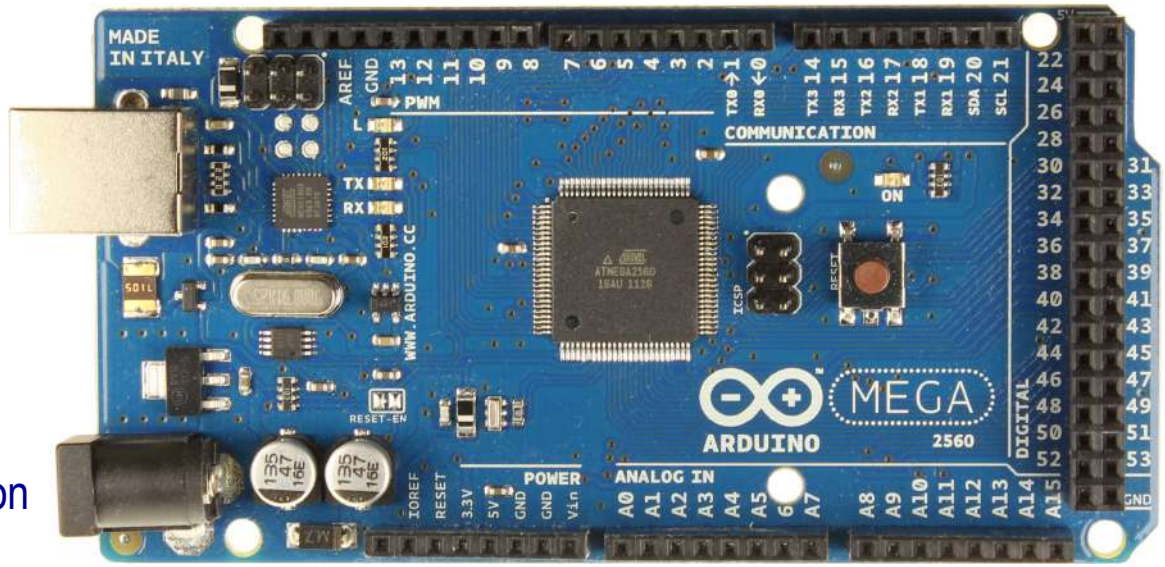
La carte Arduino Mega 2560 est une carte à microcontrôleur basée sur un **ATmega2560**

ATmega2560	256Ko (Flash)	4Ko (EEPROM)	8Ko (SRAM)	16MHz (Horloge)

14 Entrées/Sorties numériques
dont 12 sorties analogiques (PWM)

8 Entrées/Sorties numériques
dont 10 pour la communication

USB



32 Entrées/Sorties Numériques

Connecteur d'alimentation jack (7-12V)

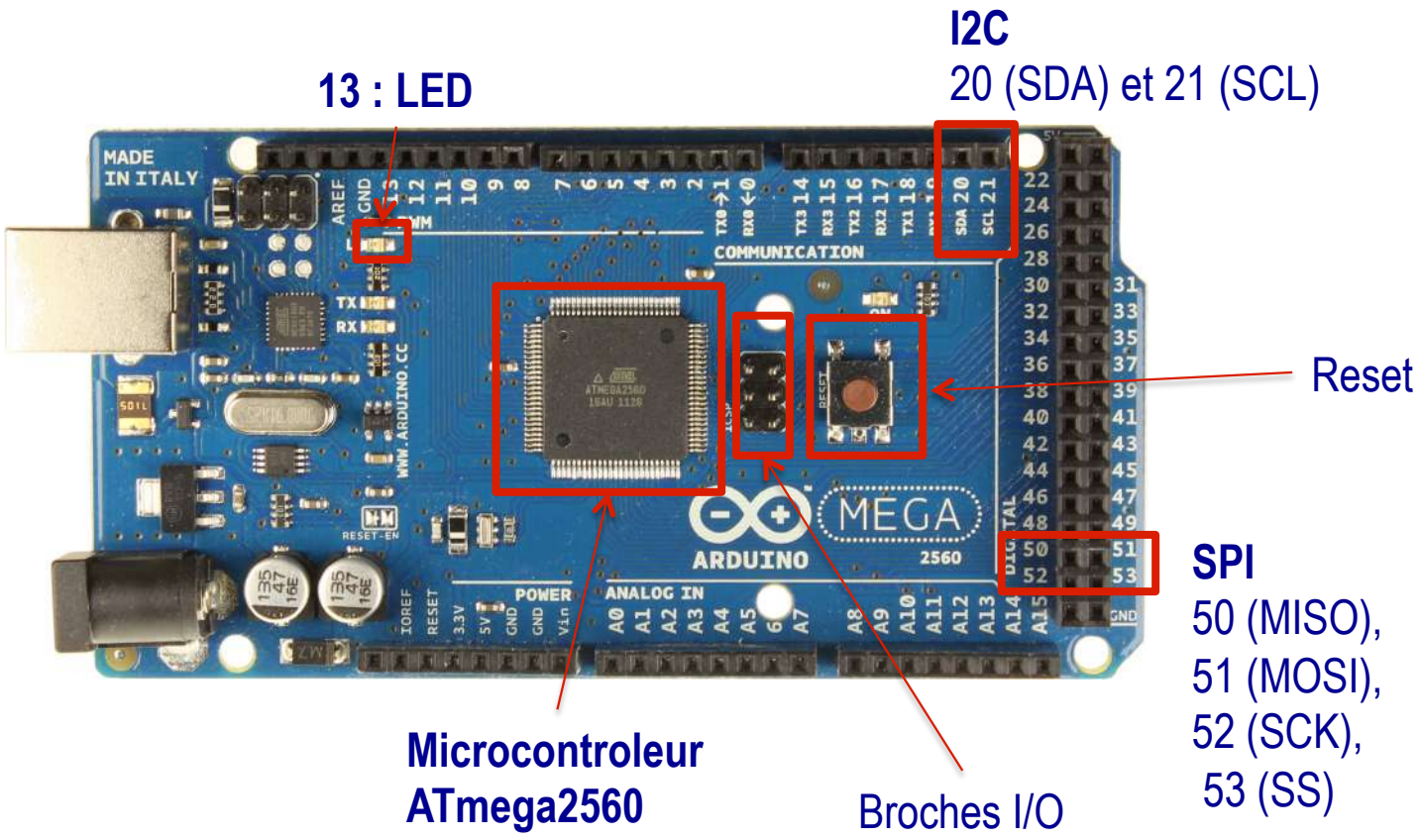
Alimentation
RESET, 3V, 5V
2 GND et 1 Vin

16 Entrées analogiques
dont 16 Entrées/Sorties numériques

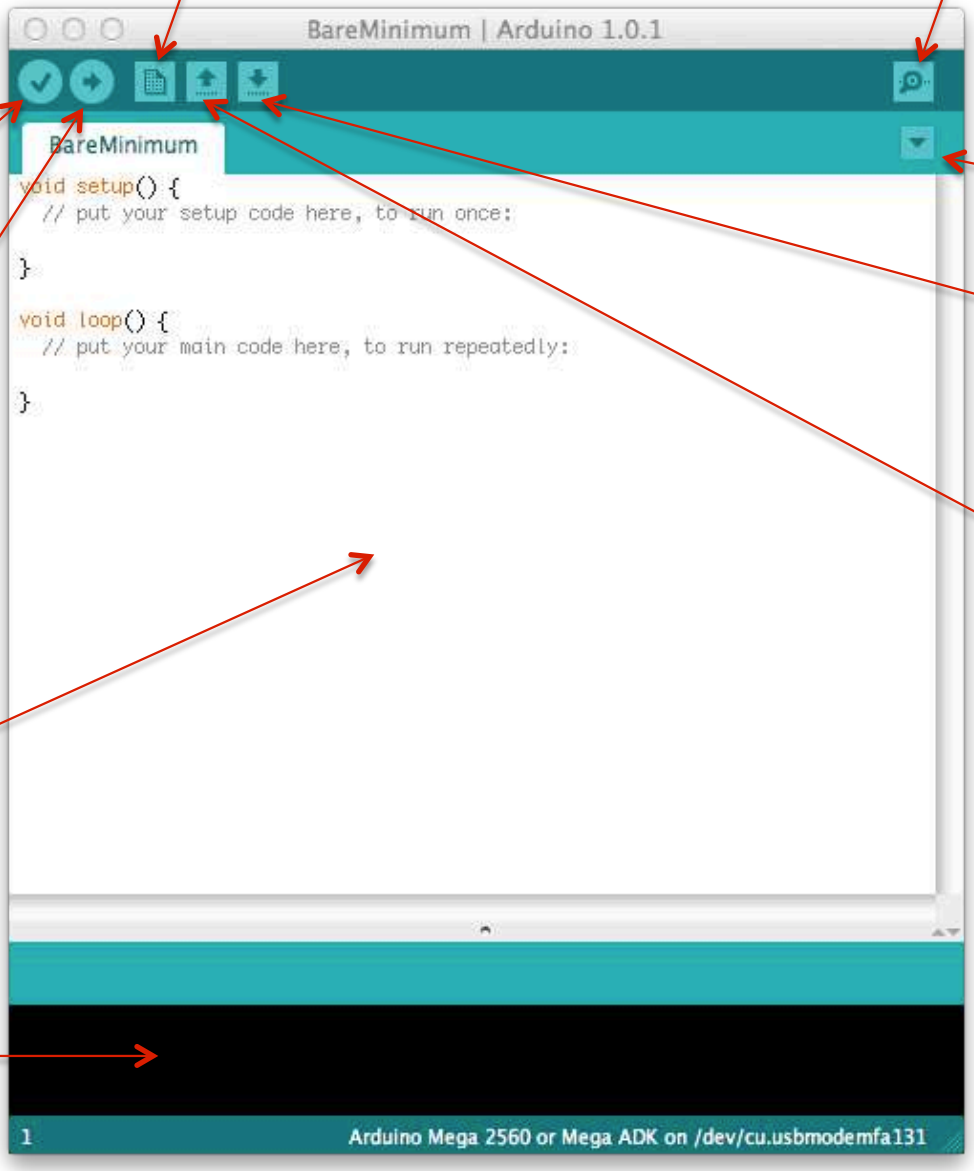
Arduino (MEGA 2560)

La carte Arduino Mega 2560 est une carte à microcontrôleur basée sur un **ATmega2560**

ATmega2560	256Ko (Flash)	4Ko (EEPROM)	8Ko (SRAM)	16MHz (Horloge)



IDE



Nouveau code

Ouvrir l'hyper-terminal

Ajouter des onglets

Ouvrir

Enregistrer

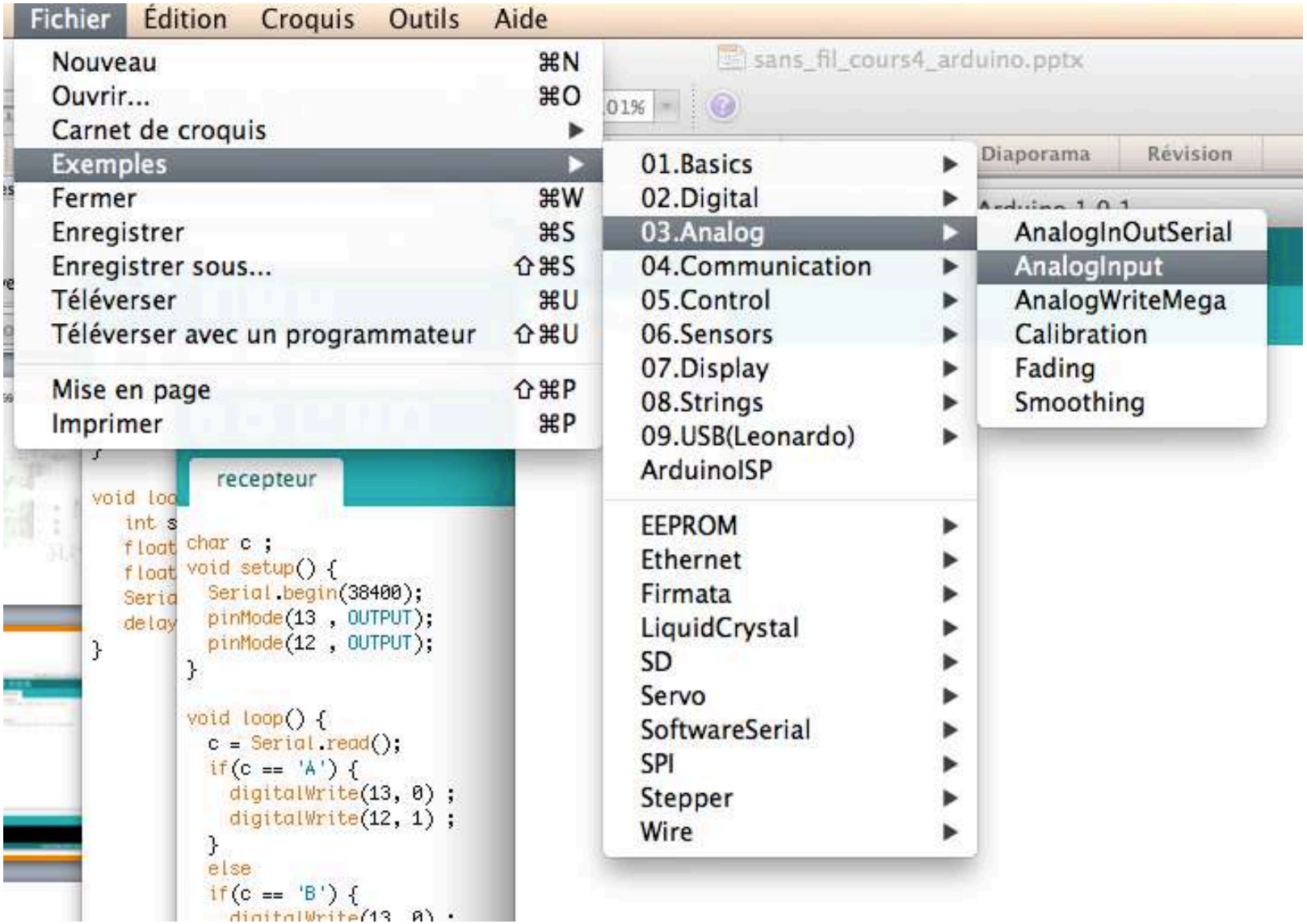
Vérifier le code

Compiler et charger le code sur la carte

Editeur de scripts

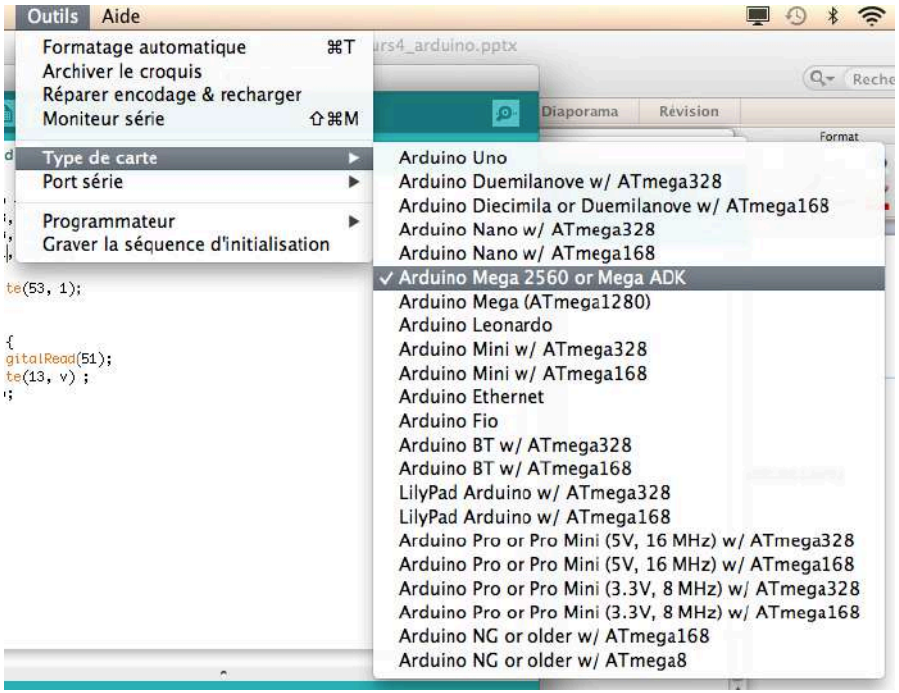
Console : Messages d'état et d'erreurs

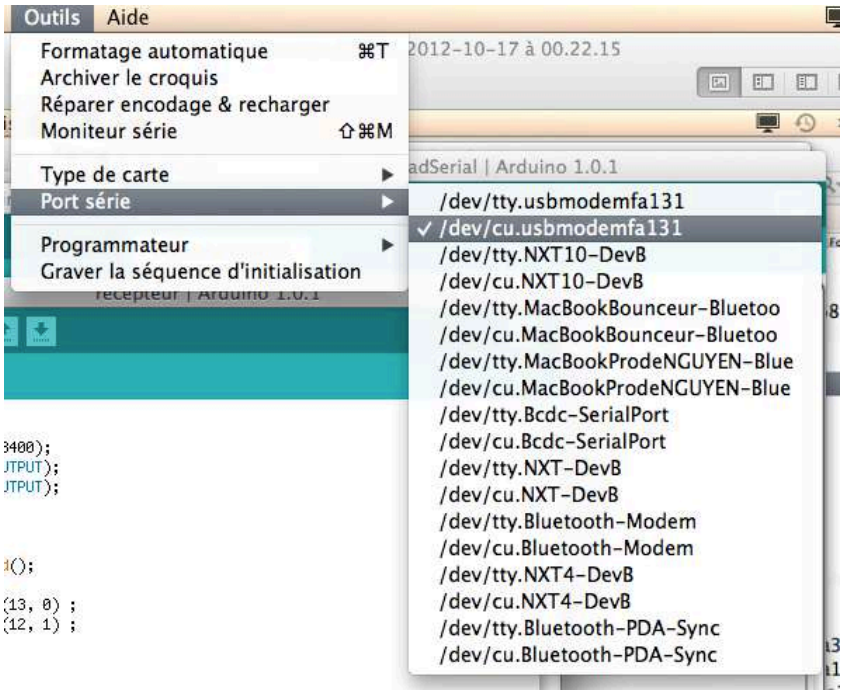
IDE : exemples de codes



Installation

- www.arduino.cc
  - Installation
- Installer les drivers si nécessaire (pour la Mega 2560, le driver n'est pas nécessaire)
- Choisissez la carte Arduino ciblée et le port utilisé





Arduino : langage

```
void setup() {
        ...
}
```

Se lance au début de l'exécution
Utile pour l'initialisation

```
void loop() {
        ...
}
```

Se lance après setup()
et en boucle infini

Arduino : langage

```
int x = 4 ;
void setup() {
        ...
}

void loop() {
        ...
}
```

On peut déclarer des variables globales

Arduino : langage

```
int x = 4 ;
void setup() {
        ...
}

void loop() {
        mafonction() ;

        ...
}
Void mafonction() {
        ...
}
```

On peut ajouter des fonctions

Arduino : langage

```
#define A 1000
int x = 4 ;
void setup() {
        ...
}

void loop() {
        mafonction() ;
        delay(A) ;
        ...
}
Void mafonction() {
        ...
}
```

Arduino : langage

- Les opérations de base :
  - Écrire dans une broche numérique (***Deigital Output***)
    - Allumer une LED, contrôler un moteur, émettre un son, etc.
  - Écrire dans une broche analogique (***Analog Output***)
    - Contrôler l'intensité d'une LED, contrôler la vitesse d'un moteur, etc.
  - Lire à partir d'une broche numérique (***Digital Input***)
    - Lire l'état d'un capteur envoyant des 0-1 : interrupteur, de mouvement, tilt, etc.
  - Lire à partir d'une broche analogique (***Analog Input***)
    - Lire la valeur d'un capteur envoyant un signal continu : capteur de lumière, de gaz, de température, d'humidité, etc.
  - Communication série (***Serial Communication***)
    - Communiquer avec un ordinateur, avec une autre carte, échanger des données, etc.

Arduino : langage

- **Symboles spéciaux :**
  - Point virgule `;`
    - Marquer la fin d'une instruction
  - Accolades `{}`
    - Pour marquer les blocs
  - Commentaires :
    - `//`
    - `/* … */`

Arduino : langage

- **Constantes et Variables :**
  - `const int X = 3 ;`
  - `boolean b = true ;` // ou false (aussi 0 ou 1) → booléen
  - `char c = 'A' ;` // ou 65 (valeurs de -128 à 127) → caractère
  - `byte oc = 60 ;` // octet : valeurs de 0 à 255
  - `int i = 487 ;` // entier (valeurs de -32768 à 32767)
  - `unsigned int j = 543 ;` // entier non signé (valeurs de 0 à 65535)
  - `long k = 65069 ;` // entier long (valeurs de -2147483648 à 2147483647)
  - `unsigned long m = 56 ;` // entier long non signé (valeurs de 0 à 4294967295)
  - `float x = 12.4 ;` // réel (à utiliser avec modération) → 4 octets
  - `double y = 34.543 ;` // réel allant jusqu'à : 1.7976931348623157 x 10^{308}
  - `string s = "bonjour" ;` // chaine de caractères (équivalent à : char s[] = "bonjour" ;
    ou à char s[8] = "bonjour" ;
    et ne pas oublier, le dernier caractère = 0)
  - `int t[3] = {4, 6, 1} ;` // tableau (ici d'entiers)

Arduino : langage

- **Structures de contrôle :**
  - **if … else**

```
if (x == 1) {
     digitalWrite(LED, HIGH) ;
}
```

  - **for**

```
for(int i = 0; i < 10; i++) {
    Serial.print("salut") ;
}
```

Arduino : langage

- **Structures de contrôle :**
  - **while**

```
while(v < 512) {
    digitalWrite(13, 1) ;
    delay(100) ;
    digitalWrite(13, 0) ;
    v = v + 1 ;
}
```

  - **do … while**

```
do {
    digitalWrite(13, 1) ;
    delay(100) ;
    digitalWrite(13, 0) ;
    v = v + 1 ;
} while (v < 512) ;
```

Arduino : langage

- **Structures de contrôle :**
  - **switch case**

```
switch(v) {
    case 34 :
            digitalWrite(12, HIGH) ;
            break ;
    case 68 :
            digitalWrite(13, HIGH) ;
            break ;
    default :
            digitalWrite(12, LOW) ;
            digitalWrite(13, LOW) ;
}
```

Arduino : langage

- **Structures de contrôle :**
  - **`break ;`**
    - permet de quitter une boucle (for, while, switch, etc.)
  - **`continue ;`**
    - force à passer à l'itération suivante dans une boucle sans exécuter le code suivant (le continue)
  - **`return`**
    - permet de retourner une valeur d'une fonction

```
int triple(int x)
    return v*3 ;
}
```

Arduino : langage

- **Formules arithmétiques :**

```
+
–
*
/
%
()

x = 2 + a * 4 ;
intensite = ((12 * v) – 5) / 2 ;
reste = 5 % 2 ;
```

Arduino : langage

- **Opérateurs de comparaison :**

==	**égal**
!=	**non égal**
<	**inférieur**
>	**supérieur**
<=	**inférieur ou égal**
>=	**supérieur ou égal**

Arduino : langage

- **Opérateurs booléens :**

`&&`	**ET**
`||`	**OU**
`!`	**NON**

```
if ((capteur >= 5) && (capteur <= 10)) {
    …
}
```

Arduino : langage

- **Opérateurs composés :**

```
++
--
+=
-=
*=
/=
```

```
v++ ;       → v = v + 1 ;
v -= 4 ;    → v = v - 4 ;
```

Arduino : langage

- **Fonctions Entrées/Sorties :**
  - **pinMode(pin, mode)**

```
pinMode(7, INPUT) ;
pinMode(13, OUTPUT) ;
```

  - **digitalWrite(pin, valeur)**

```
digitalWrite(13, HIGH) ;
```

  - **int digitalRead(pin)**

```
v = digitalRead(12) ;
```

  - **analogWrite(pin, valeur)**

```
analogWrite(9, 128) ;
```

  - **int analogRead(pin)**

```
x = analogRead(0) ;
```

Arduino : langage

- **Fonctions Entrées/Sorties :**
  - **shiftOut(dataPin, clockPin, bitOrder, value)**
    - **Pour envoyer des données à un registre à décalage.**
    - **dataPin : la broche des données,**
    - **clockPin : la broche de l'horloge,**
    - **bitOrder : l'ordre des octets (LSBFIRST, MSBFIRST)**
    - **value : l'octet à envoyer.**

```
shiftOut(2, 3, LSBFIRST, 255) ;
```

Arduino : langage

- **Fonctions Temps :**
  - **delay(ms)**
    - **Une pause de ms millisecondes.**

```
delay(500) ;
```

  - **delayMicroseconds(us)**
    - **Une pause de us microsecondes.**

```
delayMicroseconds(1000) ;
```

  - **unsigned long millis()**
    - **Retourne le nombre de millisecondes passées depuis l'exécution,**

```
duree = millis() – t ; // le temps passé depuis t
```

Arduino : langage

- **Fonctions Mathématiques :**
  - `min(x, y)`
  - `max(x, y)`
  - `abs(x)`
  - `constrain(x, a, b)`
    - **Retourne la valeur de x si elle est entre a et b sinon, il retourne a si x<a et b si x>b**

Arduino : langage

- **Fonctions Mathématiques :**
  - `double pow(v, puissance)`
  - `double sqrt(x)`
  - `double sin(r)`
  - `double cos(r)`
  - `double tan(r)`

Arduino : langage

- **Fonctions Nombres aléatoires :**
  - **randomSeed(seed)**
  - **long random(max)**
  - **long random(min, max)**

```
randomSeed(analogRead(5)) ;
random(10) ;                    // VA entre 0 et 9
random(4,9) ;                   // VA entre 4 et 8
```

Arduino : langage

- **Communication série :**
  - **Serial.begin(vitesse)** **(vitesse en baud)**

```
Serial.begin(9600) ;
```

  - **Serial.print(donnees)**

```
Serial.print(75) ;          // envoie 75
```

  - **Serial.print(donnees, encodage)**
    - `Serial.print(75, DEC) ;   // envoie 75`
    - `Serial.print(75, HEX) ;   // envoie 4B`
    - `Serial.print(75, OCT) ;   // envoie 113`
    - `Serial.print(75, BIN) ;   // envoie 01001011`
    - `Serial.print(75, BYTE) ;  // envoie "K"`
  - **Serial.println(donnees) : même que Serial.print(donnees) + "\r\n"**
  - **Serial.println(donnees, encodage) : Serial.print(donnees, encodage) + "\r\n"**

Arduino : langage

- **Communication série :**
  - **int Serial.available()**
    - **Renvoie le nombre d'octets non lus qui sont disponibles dans le port série.**

```
int n = Serial.available() ;
```

  - **int Serial.read()**
    - **Lis un octet.**

```
int v = Serial.read() ;
```

  - **Serial.flush()**
    - **Vider le tampon (buffer).**

```
Serial.flush() ;
```

Arduino : langage

- **Complément : on peut utiliser**

  #define a 12 **(pas de point virgule à la fin)**

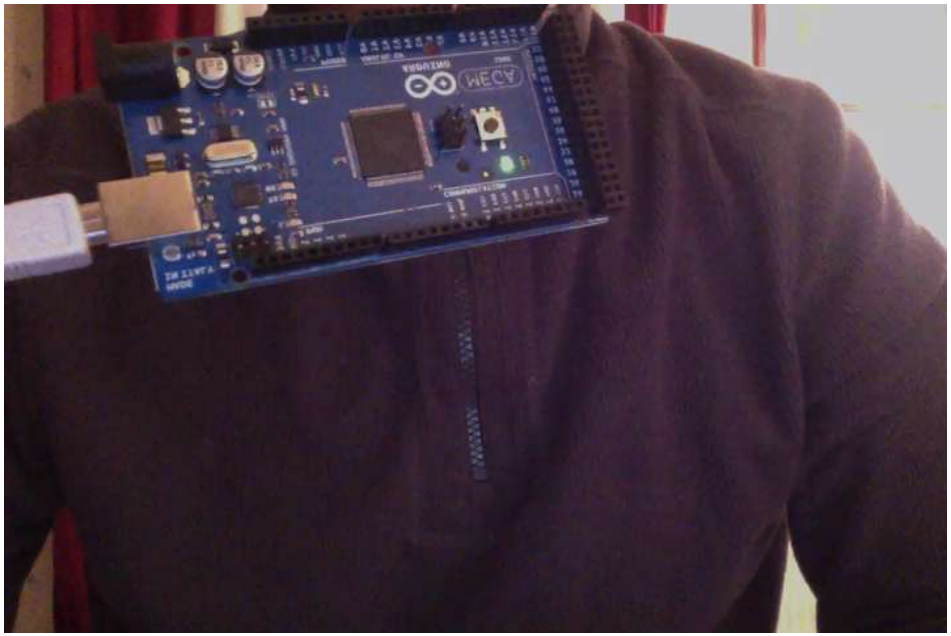
  byte b = B011010101 ;

EXEMPLES

Allumer la led de la carte (Pin n° 13)

PREMIER PROGRAMME

```
void setup() {
  pinMode(13, OUTPUT);
  digitalWrite(13, 1) ;
}

void loop() {
}
```

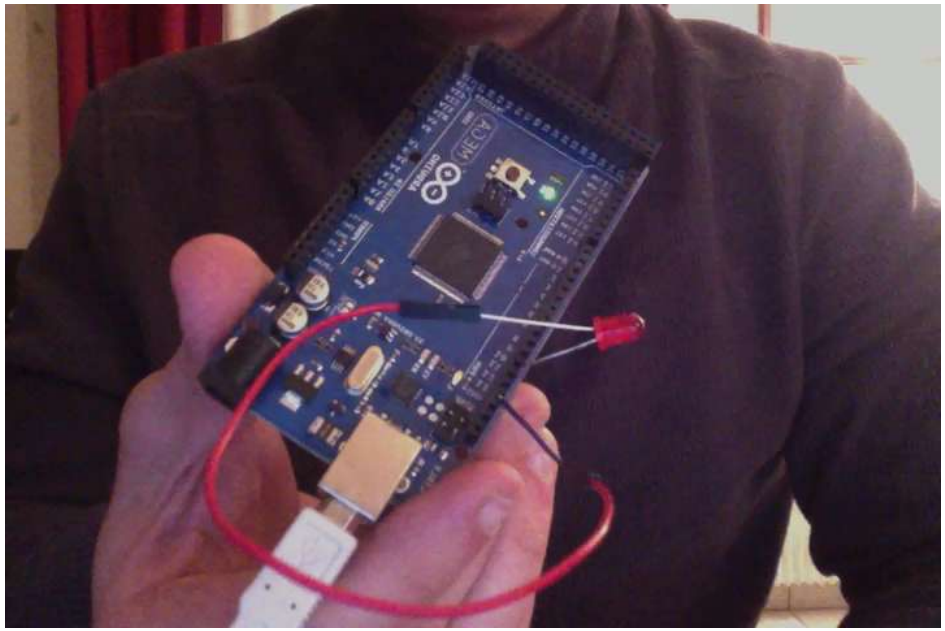


Allumer la led de la carte (Pin n° 13) : Blink

```
const int LED = 13 ;
void setup() {
  pinMode(LED, OUTPUT);
}

void loop() {
  digitalWrite(LED, HIGH) ;
  delay(1000);
  digitalWrite(LED, LOW) ;
  delay(1000);
}
```

Allumer une Led (branchée sur le pin 12)

```
void setup() {
  pinMode(12, OUTPUT);
}

void loop() {
  digitalWrite(12, 1) ;
  delay(1000);
  digitalWrite(12, 0) ;
  delay(1000);
}
```



Allumer une Led (pin 12 en analogique)

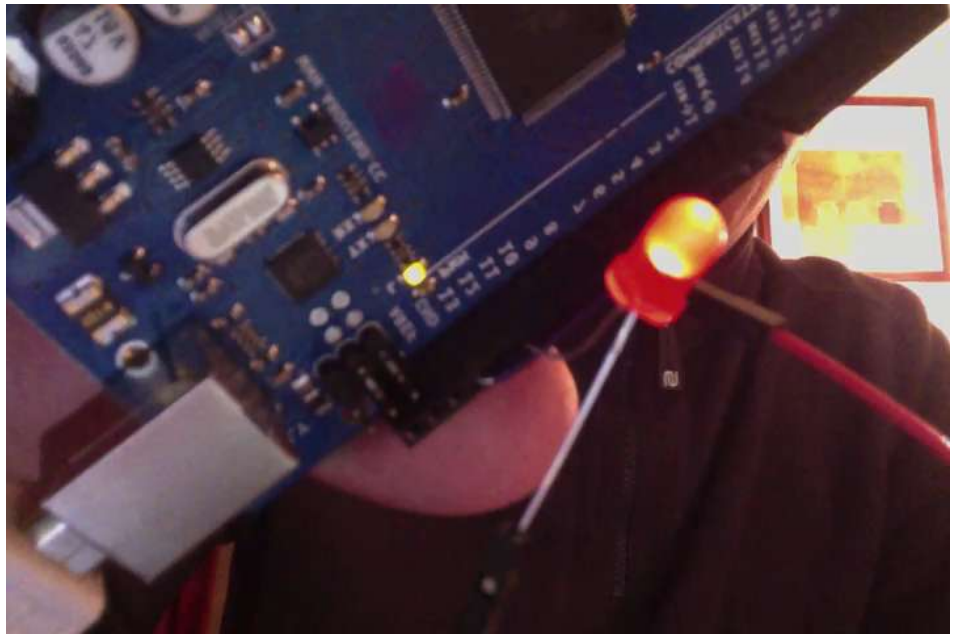
```
void setup() {
  pinMode(12, OUTPUT);
}

void loop() {
  analoglWrite(12, 54) ;
  delay(1000);
  analoglWrite(12, 0) ;
  delay(1000);
}
```

Allumer une Led (pin 12 en analogique 2)

```
void setup() {
  pinMode(12, OUTPUT);
}

void loop() {
  for(int i=0; i<20; i++) {
    analogWrite(12, i) ;
    delay(100);
  }
}
```

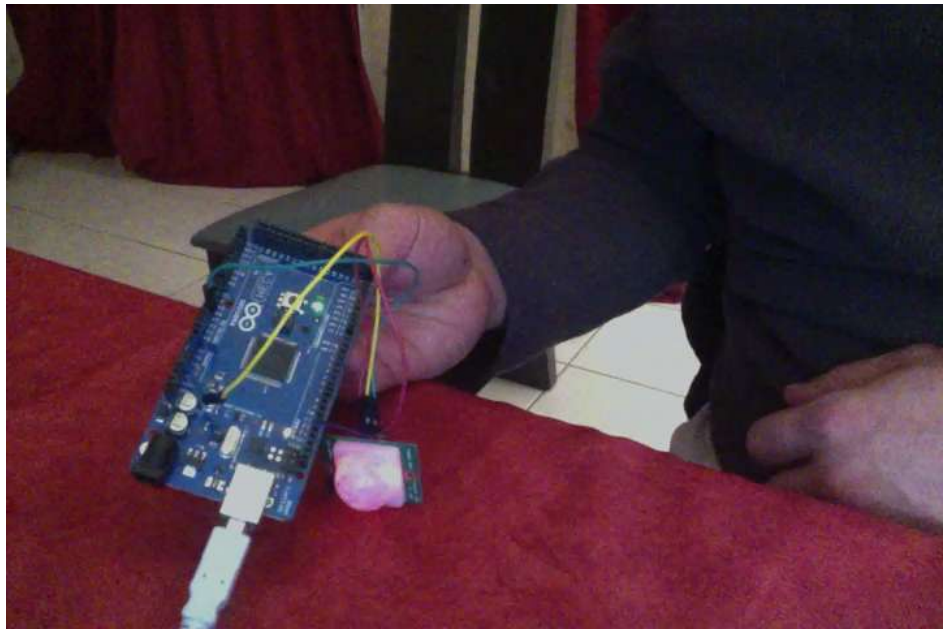
Allumer une Led (pin 12 en analogique 3)

```
int pin=11;
void setup() {
  pinMode(pin, OUTPUT);
}

void loop() {
  for(int i=0; i<20; i++) {
    analogWrite(pin, i) ;
    delay(100);
  }
  for(int i=19; i>0; i--) {
    analogWrite(pin, i) ;
    delay(100);
  }
}
```



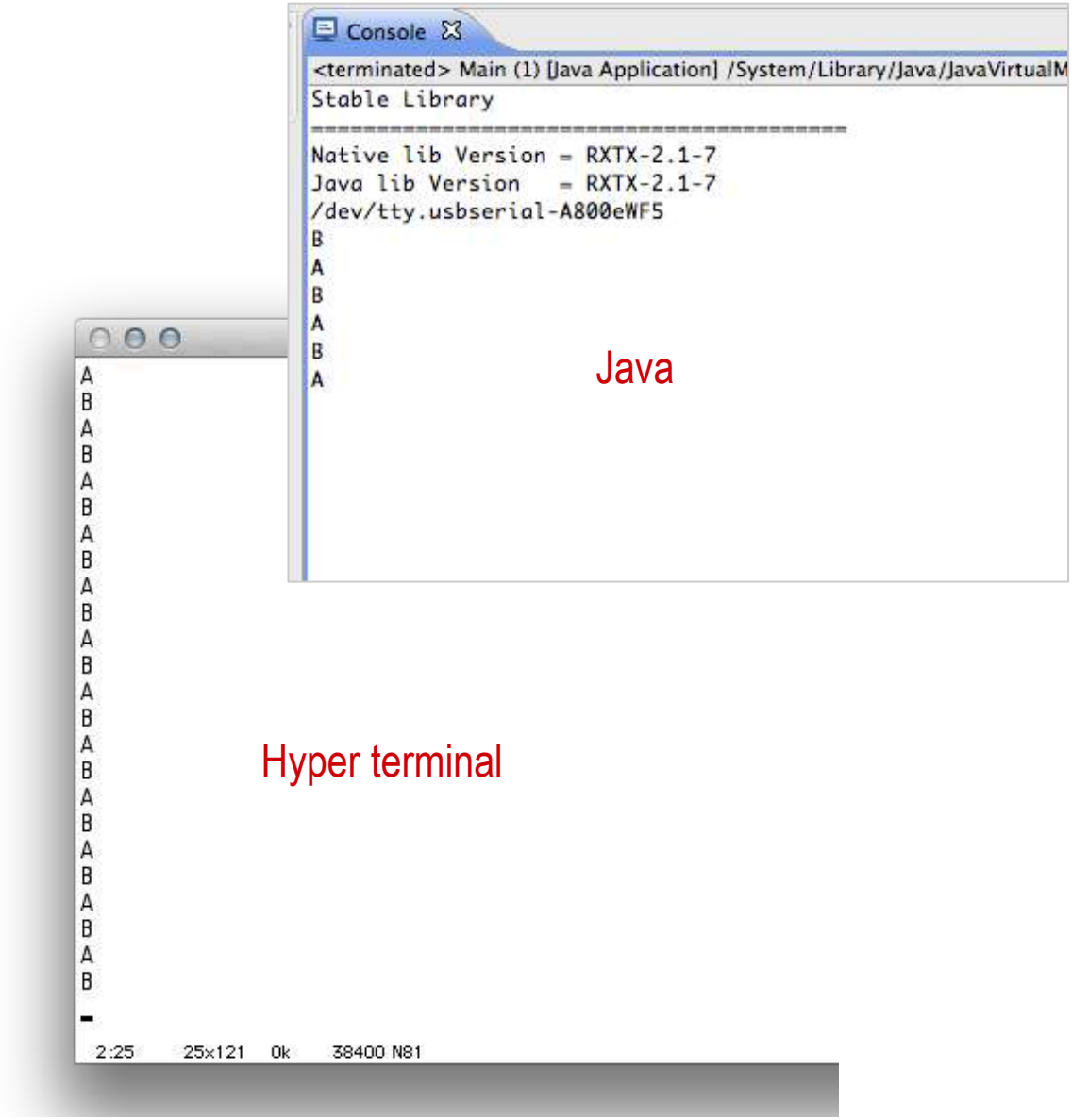
Lire la valeur d'un capteur (de mouvements)

```
int v ;
void setup() {
    pinMode(12, INPUT);
    pinMode(13, OUTPUT);
}

void loop() {
    v = digitalRead(12) ;
    digitalWrite(13, v) ;
    delay(1000);
}
```



L'émission

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.println('A') ;
  delay(1000);
  Serial.println('B') ;
  delay(1000);
}
```



La réception

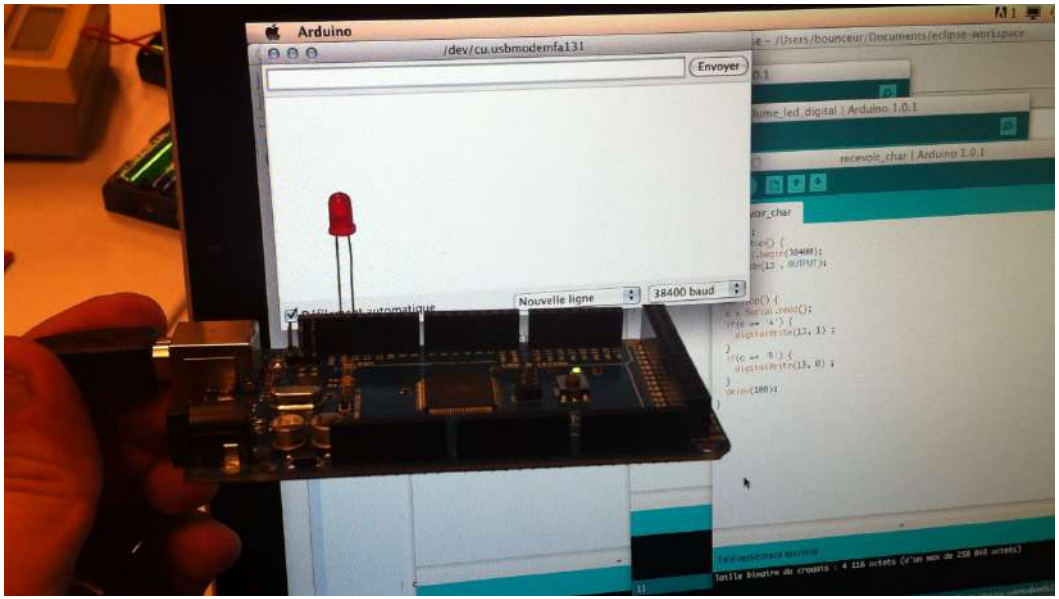
```
char c ;
void setup() {
  Serial.begin(9600);
}

void loop() {
  c = Serial.read() ;
  Serial.println(c) ;
  delay(1000);
}
```

La réception

```
char c ;
void setup() {
  Serial.begin(9600);
}

void loop() {
  if(Serial.available()>0) {
    c = Serial.read() ;
    Serial.println(c) ;
  }
}
```

La réception (allumer/éteindre une Led à distance)

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

void loop() {
 if(Serial.available()>0) {
  c = Serial.read();
  if(c == 'A') {
    digitalWrite(13, 1) ;
  }
  if(c == 'B') {
    digitalWrite(13, 0) ;
  }
  delay(1000);
 }
}
```



Émission et Réception entre deux cartes Arduino

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.write('A') ;
  delay(1000);
  Serial.write('B') ;
  delay(1000);
}
```

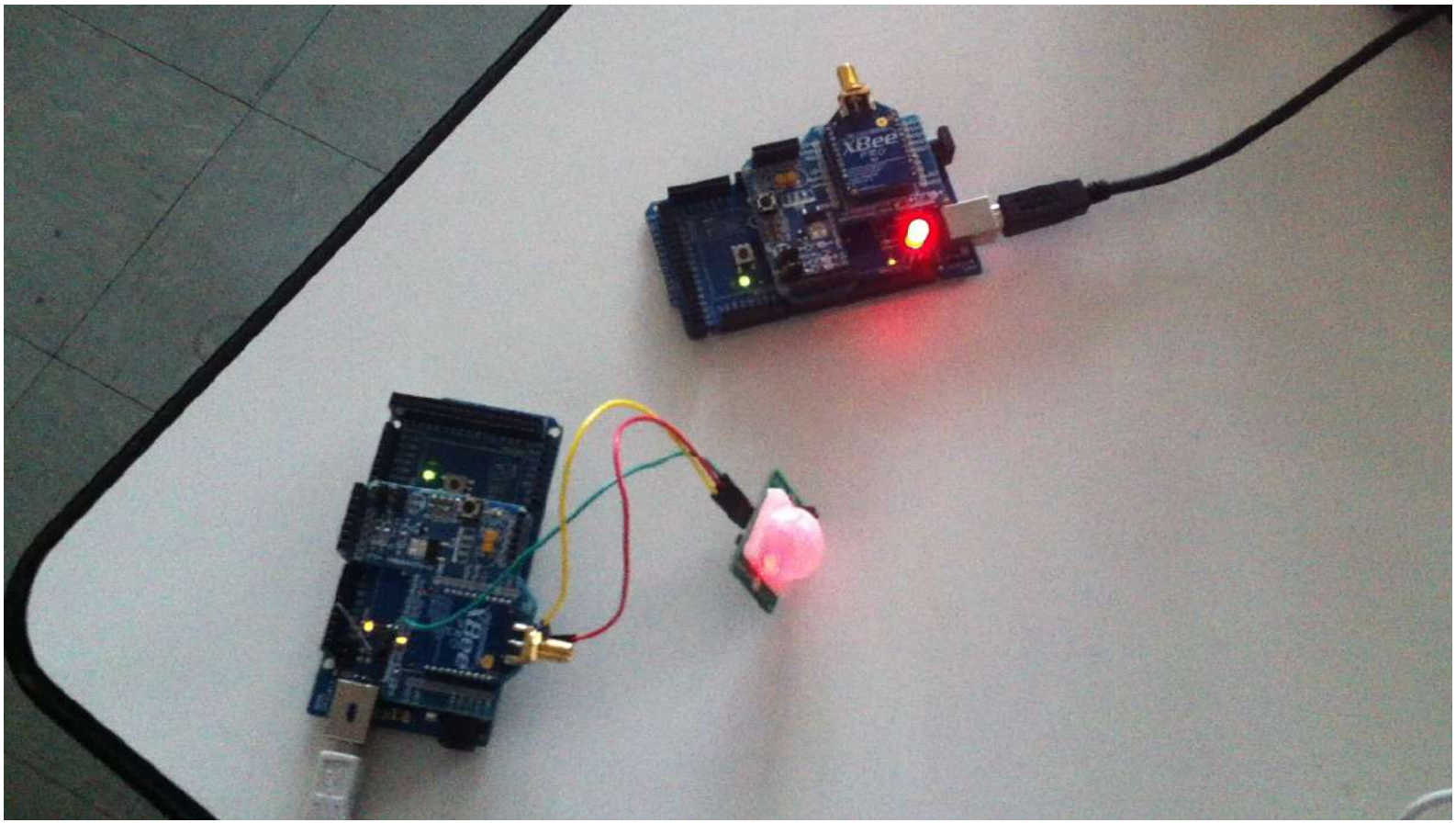
```
char c ;
void setup() {
  Serial.begin(9600);
  pinMode(13 , OUTPUT);
}
void loop() {
 if(Serial.available()>0) {
  c = Serial.read();
  if(c == 'A') {
    digitalWrite(13, 1) ;
  }
  if(c == 'B') {
    digitalWrite(13, 0) ;
  }
 }
}
```

Émission et Réception (capteur de mouvements)

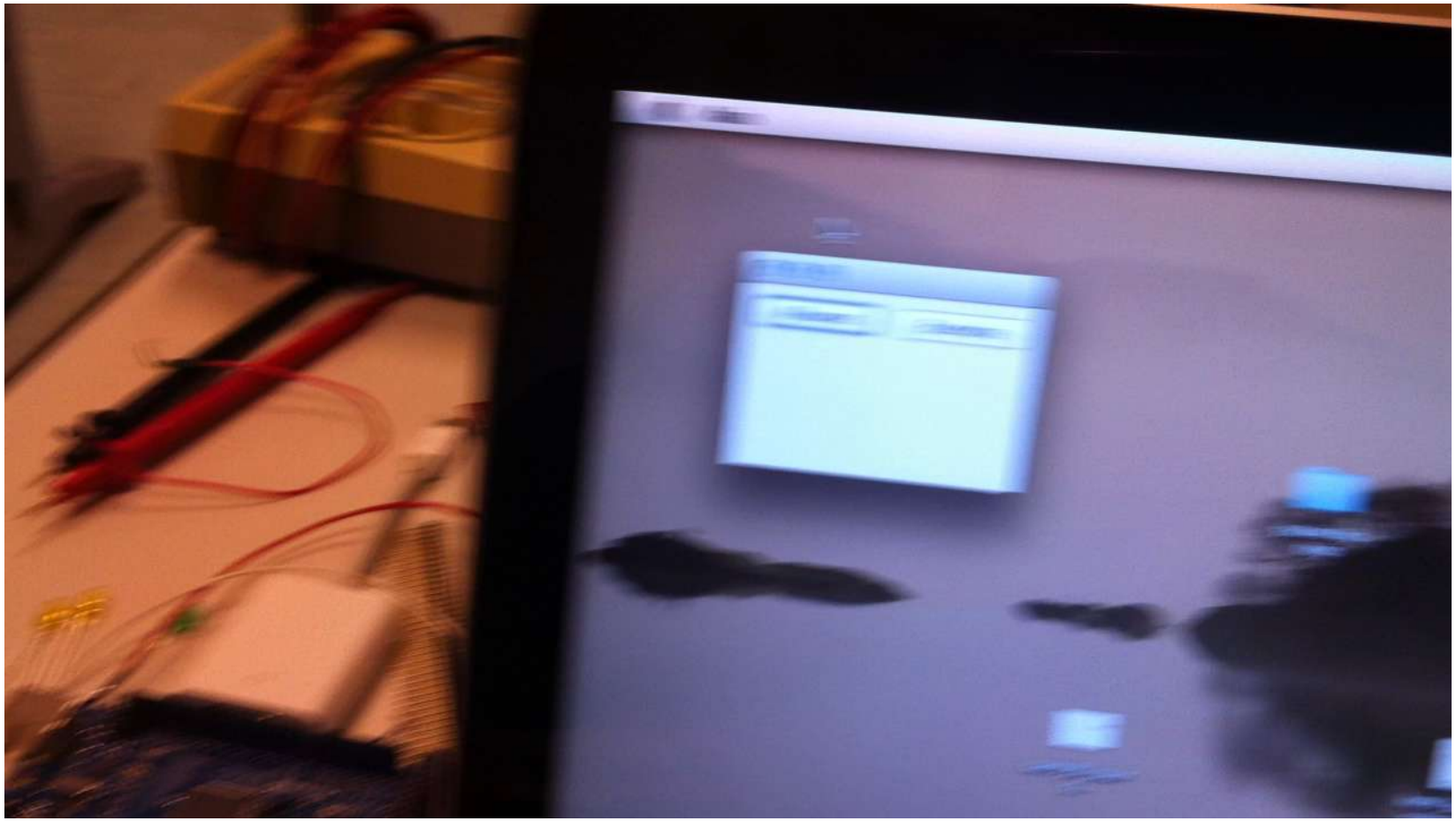
```
int v ;
void setup() {
  Serial.begin(9600);
  pinMode(12, INPUT);
}
void loop() {
  v = digitalRead(12);
  if(v==1)
    Serial.write('A') ;
  if(v==0)
    Serial.write('B') ;
  delay(100);
}
```

```
char c ;
void setup() {
  Serial.begin(9600);
  pinMode(13 , OUTPUT);
}
void loop() {
  c = Serial.read();
  if(c == 'A') {
    digitalWrite(13, 1) ;
  }
  if(c == 'B') {
    digitalWrite(13, 0) ;
  }
  delay(100);
}
```

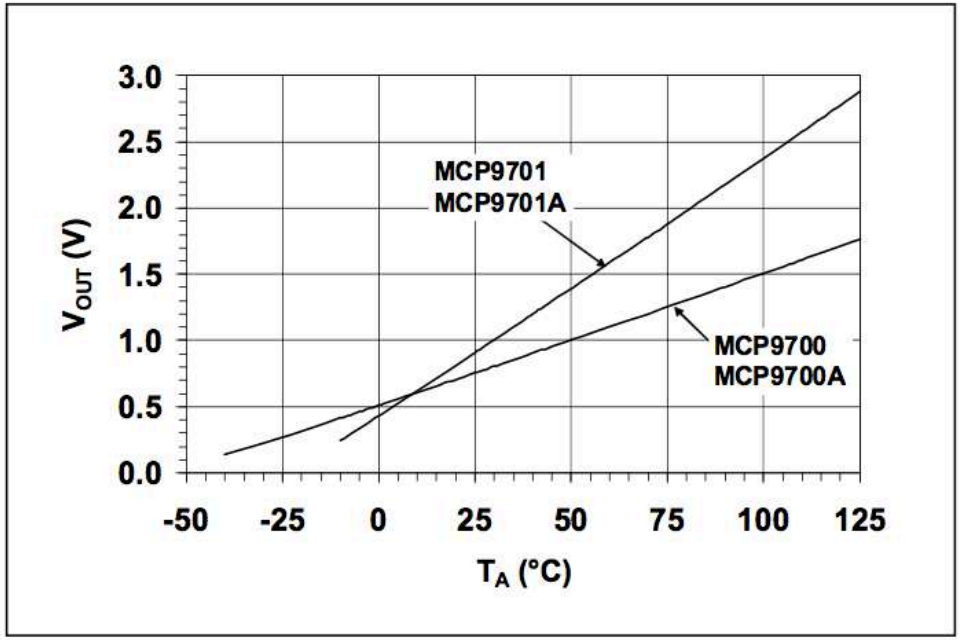
Émission et Réception (capteur de mouvements)



TP à faire : Émission et Réception (RXTX-Arduino)

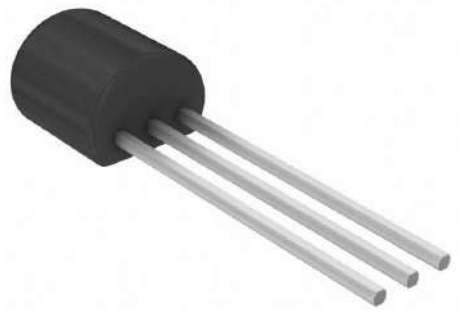


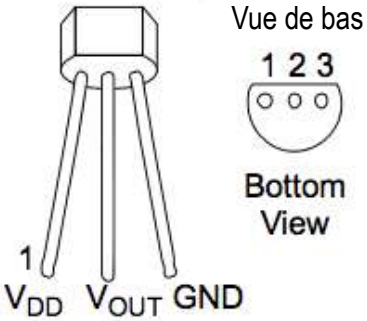
Lire la valeur d'un capteur (analogique)




```
void setup() {
    Serial.begin(9600);
}

void loop() {
    int sensorValue = analogRead(A0);
    float voltage = sensorValue * (5.0 / 1023.0);
    float temperature = 100*voltage-50;
    Serial.println(temperature);
    delay(1000);
}
```

$y = 100x - 50$





Merci