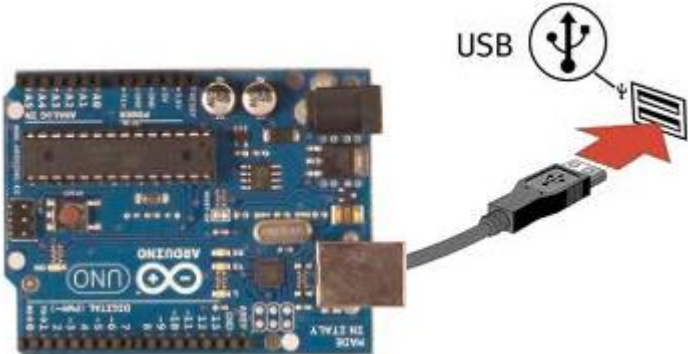
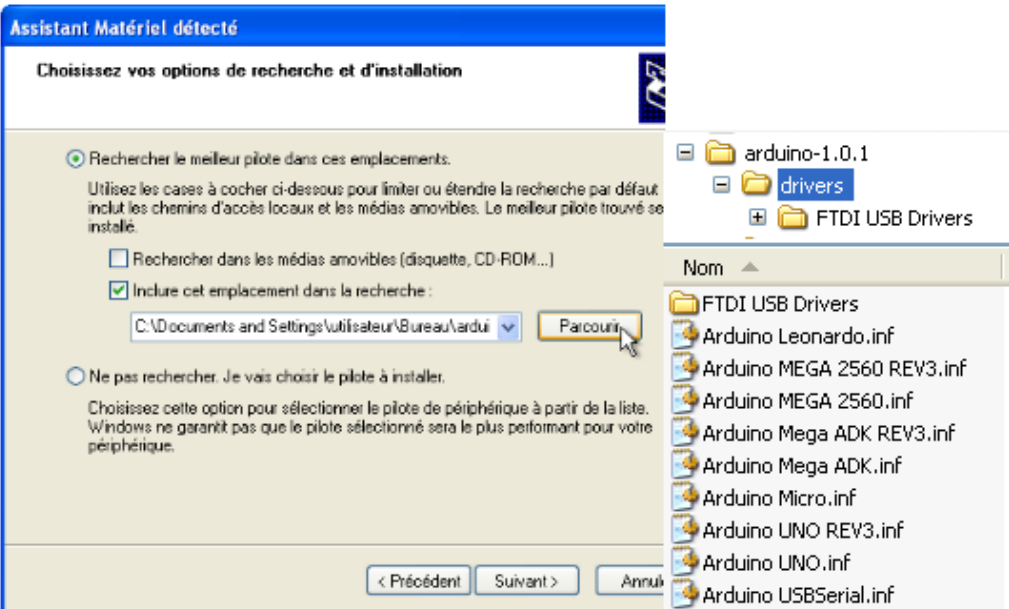
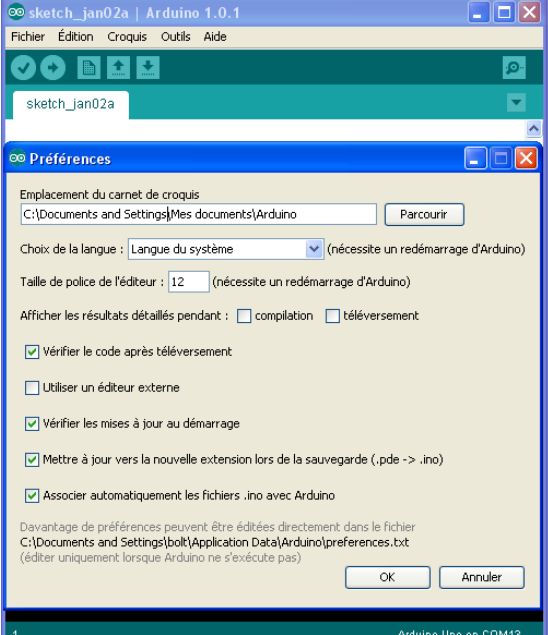

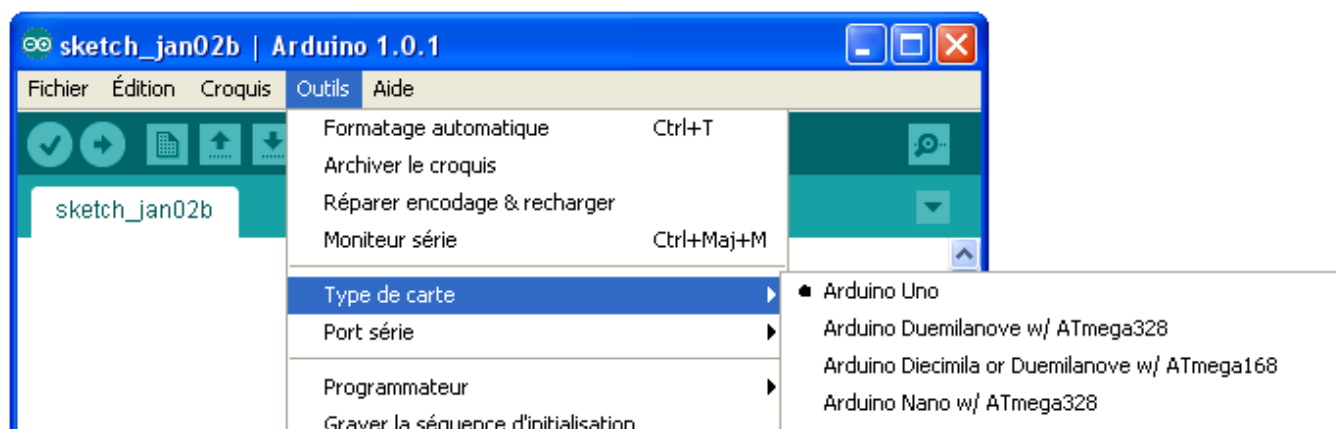


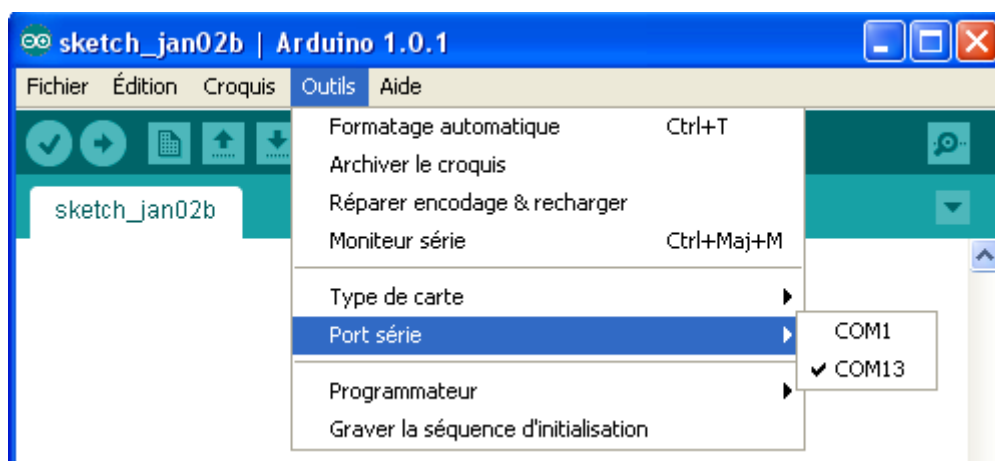
MEMO ARDUINO

Installation du pilote	
	<p>Connecter l'Arduino au PC</p>
	<p>Si c'est la 1ere connexion, effectuer une installation manuelle et pointer dans le répertoire driver du dossier arduino</p>
	<p>Lancer le logiciel  <code>arduino.exe</code> ,</p> <p>Le répertoire de travail sera :</p> <p>C:\Documents and Settings\Mes documents\Arduino</p>

Sélection de la carte Arduino UNO



Choisir le bon port de communication COM

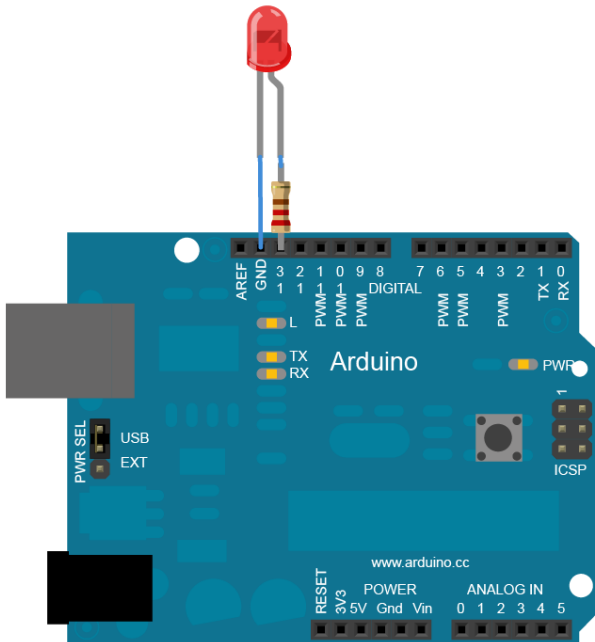


Remarque :

- Le port COM1 est le port série RS232 du PC ;
- Le port COM13 est celui de l'Arduino ;
- Si vous connectez un autre Arduino sur le pc, le numéro du port COM sera incrémenté)
- Il est judicieux d'affecter un Arduino par PC afin de ne pas surcharger les numéros de port COM.

Mon 1^{er} programme <http://arduino.cc/en/Tutorial/Blink>

Repérage des broches des connecteurs d'entrées/sorties

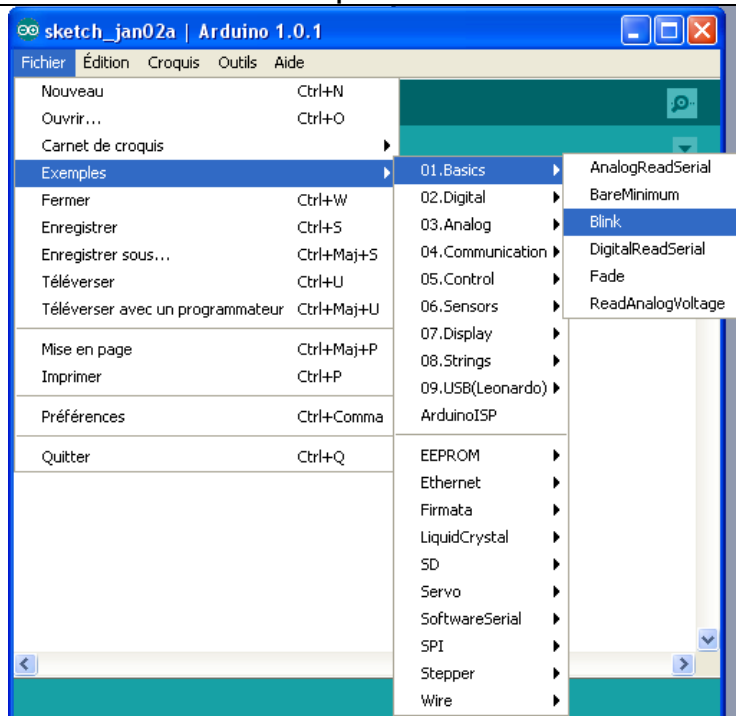


Les broches 0 à 13 sont des Entrées/Sorties logiques (0V/5V)

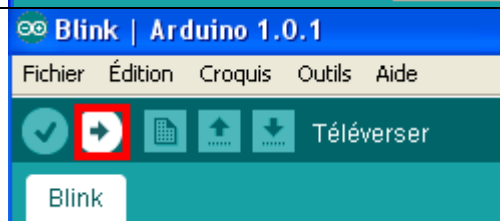
Les broches 0(RX) et 1(TX) sont réservées pour la communication entre le PC et l'Arduino.

Led L est commandé via les broches 13 et est active au niveau haut. Si on ajoute une led rouge en série avec une résistance de 220ohms comme l'indique le schéma ci contre, les 2 leds seront commandées en même temps.

Sélectionner l'exemple blink



Menu fichier,
Exemples, 01. Basics, Blink



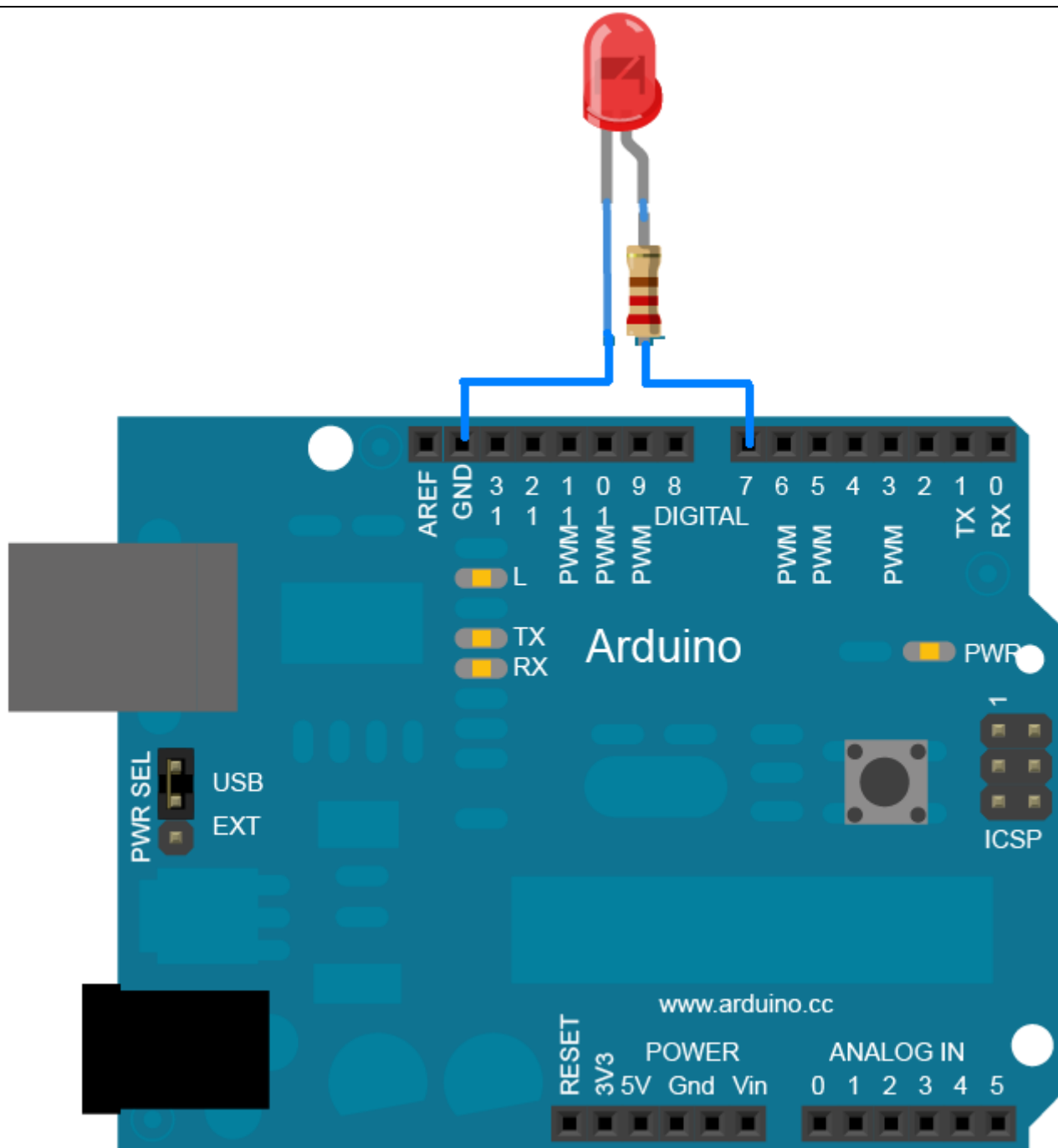
Cliquer sur Téléverser pour programmer l'Arduino avec l'exemple Blink.

Analyse du programme

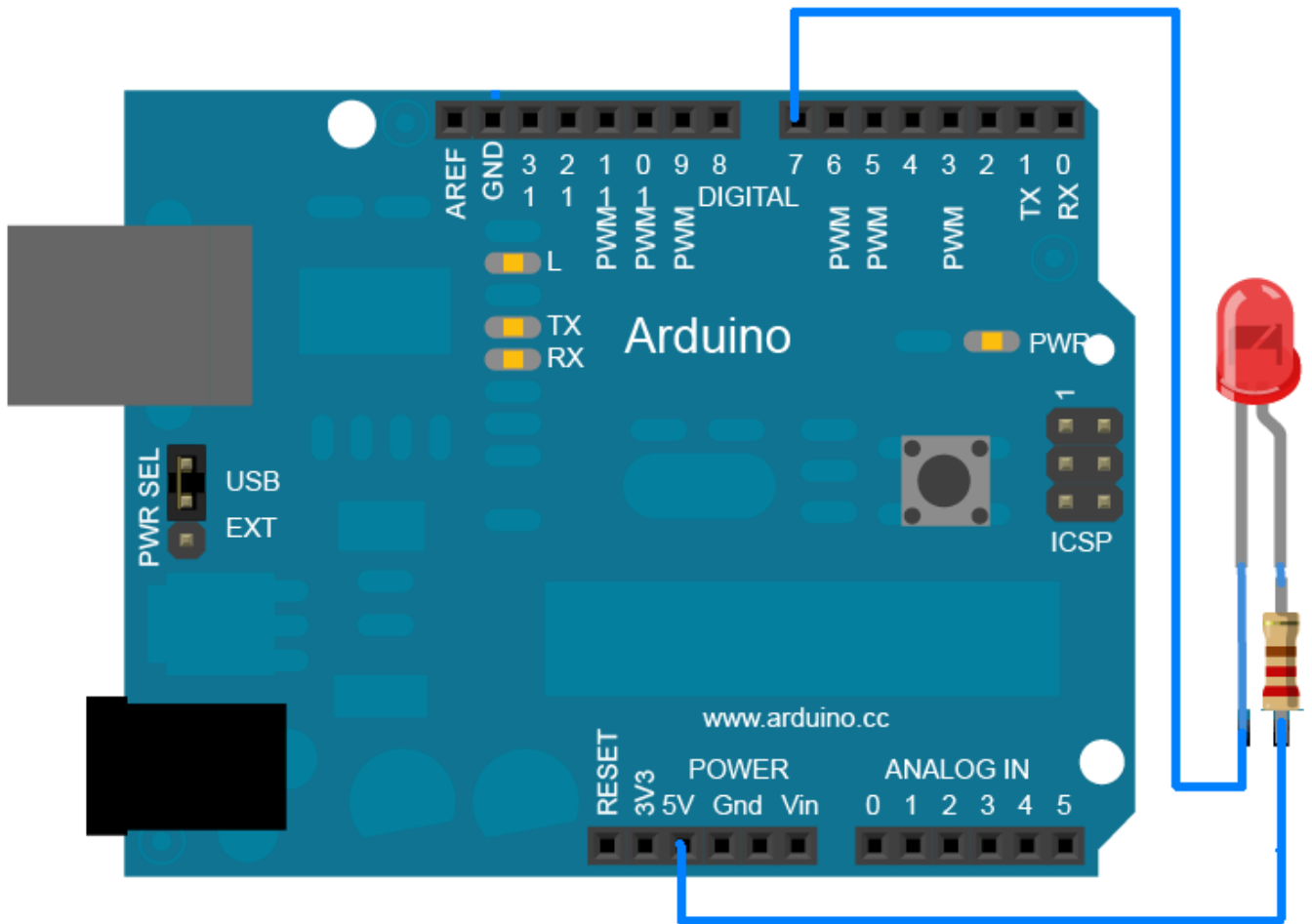
```
/*  
  Blink  
  Turns on an LED on for one second, then off for one second, repeatedly.  
  
  This example code is in the public domain.  
*/  
  
// Pin 13 has an LED connected on most Arduino boards.  
// give it a name:  
int led = 13;  
  
// the setup routine runs once when you press reset:  
void setup() {  
  // initialize the digital pin as an output.  
  pinMode(led, OUTPUT);  
}  
  
// the loop routine runs over and over again forever:  
void loop() {  
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)  
  delay(1000);             // wait for a second  
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW  
  delay(1000);             // wait for a second  
}
```

int led = 13;	Déclare une variable correspondant à la broche utilisée
pinMode(led, OUTPUT);	La broche 13 est en sortie
digitalWrite(led, HIGH);	Mise au niveau logique 1 de la sortie 13 La led est allumée
digitalWrite(led, LOW);	Mise au niveau logique 0 de la sortie 13 La led est éteinte
delay(1000);	Tempo une seconde

Modifier le programme précédent afin de faire clignoter la led rouge, mais avec la broche 7



A partir du plan de câblage suivant, Il faut que la led rouge soit tout le temps allumée. Quel sera le programme ?

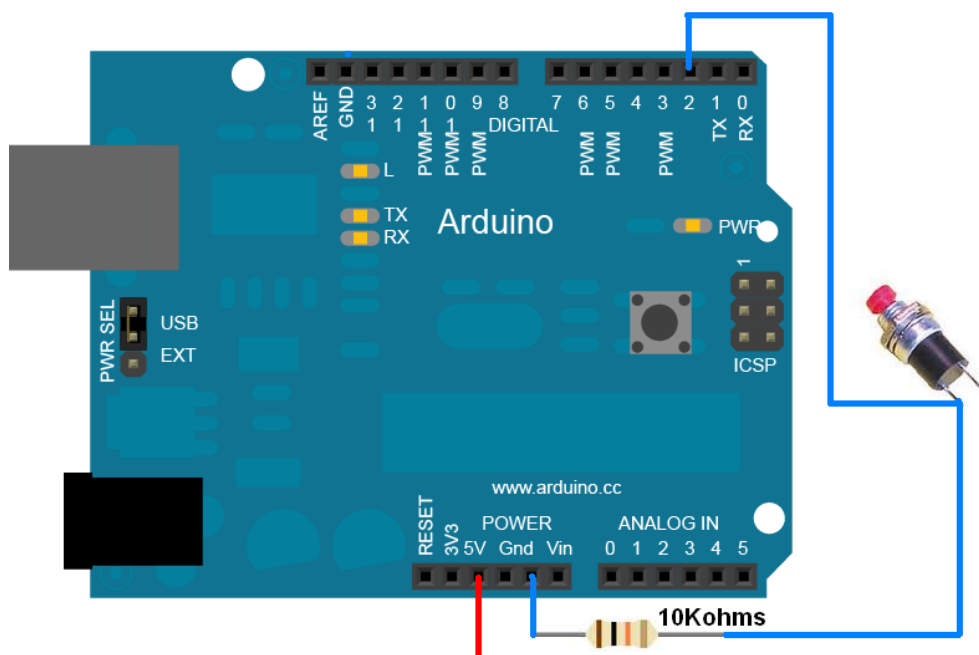


```
int led = ??;  
  
void setup() {  
  pinMode(led, OUTPUT);  
}  
  
void loop() {  
  digitalWrite(led, ??); // turn the LED on  
}
```

Expliquer le choix de l'instruction digitalWrite, par rapport au câblage de la led.

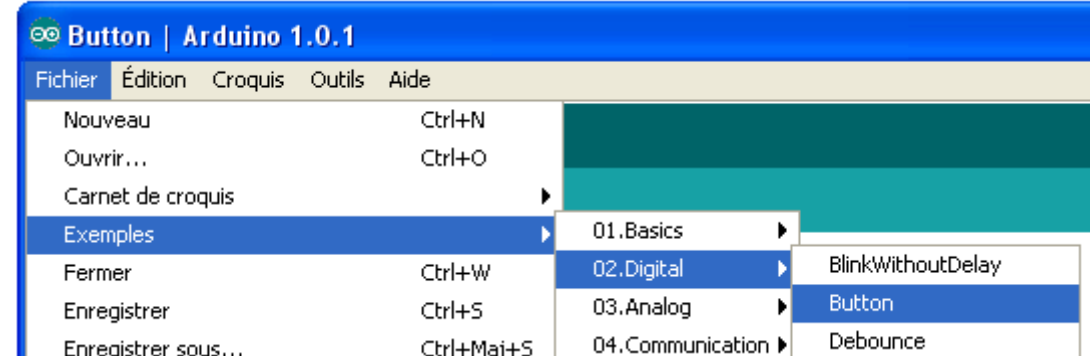
Détecter l'appuie d'un bouton poussoir <http://arduino.cc/en/Tutorial/Button>

Repérage des broches des connecteurs d'entrées/sorties

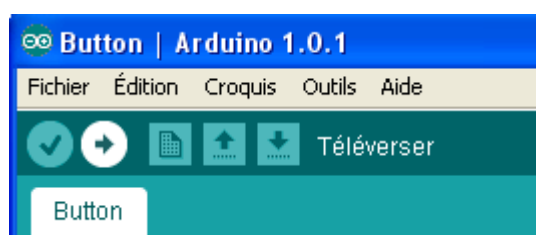


Câbler le bouton poussoir et la résistance ci-contre.

Sélectionner l'exemple button



Menu fichier, Exemples, 02. Digital, Button.



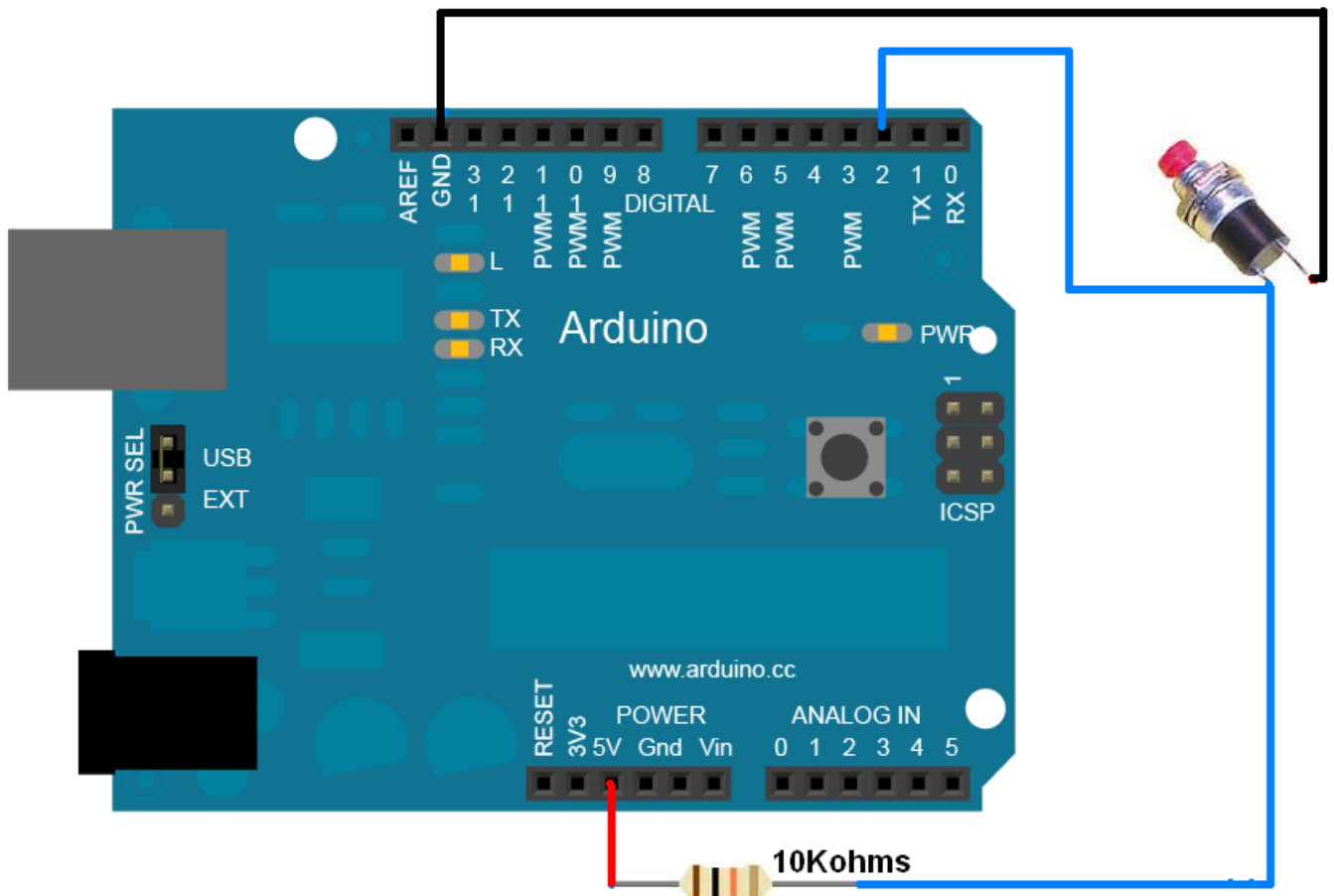
Cliquer sur Téléverser pour programmer l'Arduino avec l'exemple Button.

Analyse du programme

```
/*  
  Button  
  
  Turns on and off a light emitting diode(LED) connected to digital  
  pin 13, when pressing a pushbutton attached to pin 2.  
  
  * Note: on most Arduinos there is already an LED on the board  
  attached to pin 13.  
  
  */  
  
  // constants won't change. They're used here to  
  // set pin numbers:  
  const int buttonPin = 2;    // the number of the pushbutton pin  
  const int ledPin = 13;      // the number of the LED pin  
  
  // variables will change:  
  int buttonState = 0;        // variable for reading the pushbutton status  
  
  void setup() {  
    // initialize the LED pin as an output:  
    pinMode(ledPin, OUTPUT);  
    // initialize the pushbutton pin as an input:  
    pinMode(buttonPin, INPUT);  
  }  
  
  void loop(){  
    // read the state of the pushbutton value:  
    buttonState = digitalRead(buttonPin);  
  
    // check if the pushbutton is pressed.  
    // if it is, the buttonState is HIGH:  
    if (buttonState == HIGH) {  
      digitalWrite(ledPin, HIGH); // turn LED on:  
    }  
    else {  
      digitalWrite(ledPin, LOW);  // turn LED off:  
    }  
  }  
}
```

<code>int led = 13;</code>	Déclare une variable correspondant à la broche utilisée
<code>pinMode(buttonPin, INPUT);</code>	La broche 2 est en entrée
<code>buttonState = digitalRead(buttonPin);</code>	Lit l'état du bouton poussoir et mise à jour de la variable buttonState
<code>if (buttonState == HIGH)</code>	Test de la variable buttonState

A partir du plan ci-dessous. Quel sera le programme pour obtenir le même résultat que précédemment ?

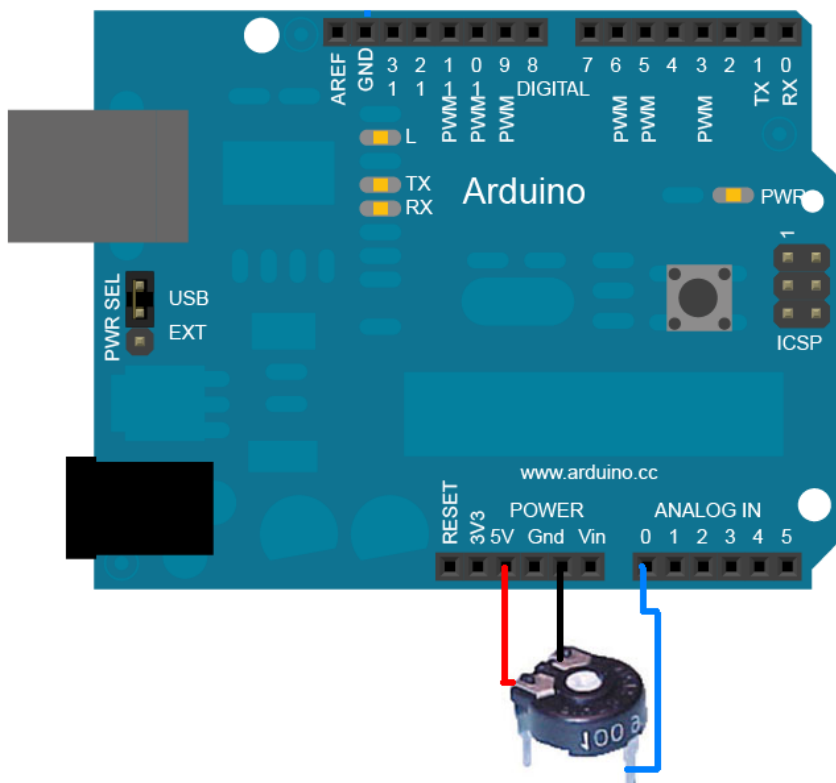


```
if (buttonState == ??) {  
    digitalWrite(ledPin, HIGH); // turn LED on:  
}  
else {  
    digitalWrite(ledPin, LOW); // turn LED off:  
}
```

Expliquer la différence entre les 2 programmes permettant de détecter l'appuie sur un bouton poussoir.

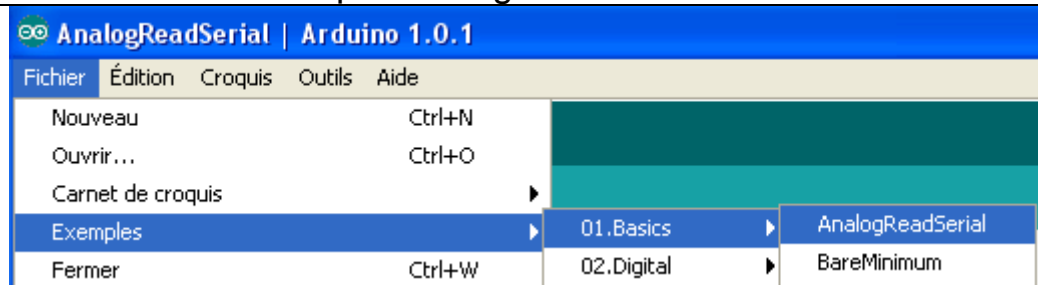
Lire une tension analogique <http://arduino.cc/en/Tutorial/AnalogReadSerial>

Repérage des broches des connecteurs d'entrées/sorties

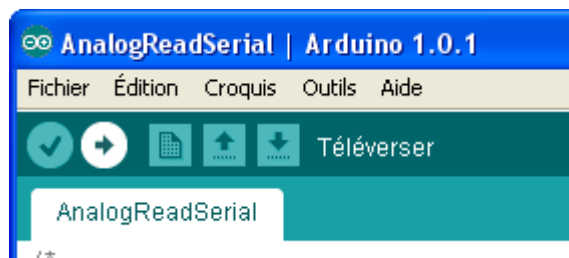


Câbler le potentiomètre ci-contre.

Sélectionner l'exemple AnalogReadSerial



Menu fichier, Exemples, 03. Analog, AnalogRead Serial.



Cliquer sur Téléverser pour programmer l'Arduino avec l'exemple AnalogRead Serial.

Analyse du programme AnalogReadSerial

```

/*
  AnalogReadSerial
  Reads an analog input on pin 0, prints the result to the serial monitor.
  Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V
  and ground.

  */

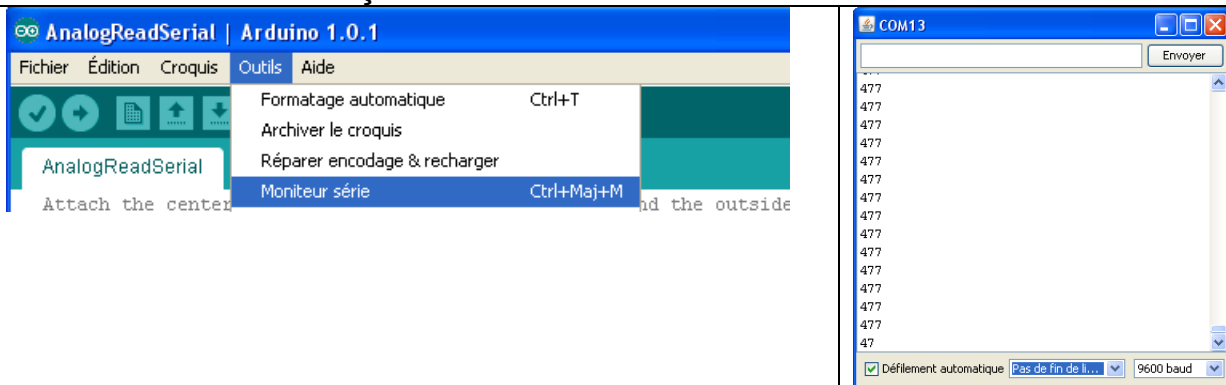
// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1); // delay in between reads for stability
}

```

Serial.begin(9600);	Initialise la liaison série
int sensorValue = analogRead(A0);	Lire la tension analogique et mise à jour de la variable sensorValue
Serial.println(sensorValue);	Envoie la valeur de sensorValue sur la liaison série en ASCII.

Vérifier les valeurs reçues avec le moniteur série :



Modifier le programme précédent (instructions en rouge)

```

/*
  AnalogReadSerial
  Reads an analog input on pin 0, prints the result to the serial monitor.
  Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V
  and ground.
  */

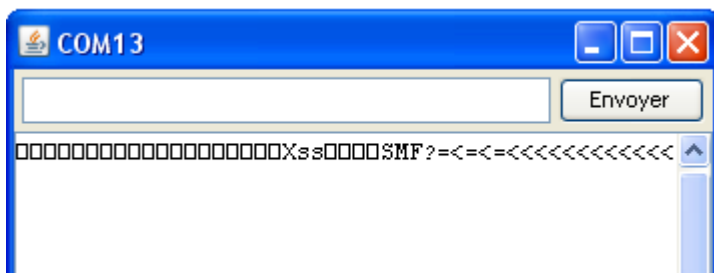
// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  // Serial.println(sensorValue);
  Serial.write(sensorValue/4);
  delay(500);
}

```

L'instruction `Serial.write(sensorValue/4)` va envoyer sur la ligne série des valeurs binaires comprises entre 0 et 255.

Les valeurs de la ligne série sont difficilement identifiables dans le moniteur.



Récupération des données séries avec Processing

```
import processing.serial.*;
PrintWriter output;

Serial myPort; // Create object from Serial class
int val;        // Data received from the serial port
int posX;

void setup()
{
    size(255, 255);
    String portName = Serial.list()[1];
    myPort = new Serial(this, portName, 9600);
    output = createWriter("positions.txt");
    fill(255,0,0);
    posX=0;
}

void draw()
{
    if ( myPort.available() > 0) { // If data is available,
        val = myPort.read();        // read it and store it in val
        println(val);
        ecritureFichier(val);
        //point (posX,val);
        ellipse(posX, val, 10, 10);
        posX++;
    }
}

void ecritureFichier(int val){
    output.println(val);
}

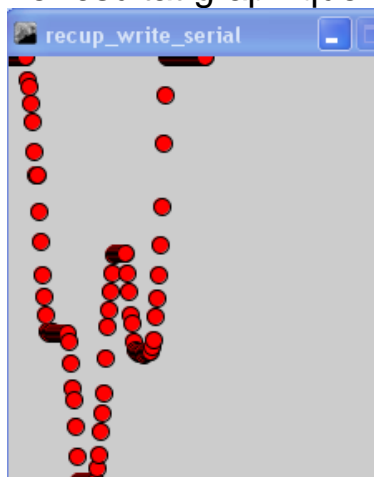
void dispose()
{
    println("Fermeture du programme");
    output.flush(); // Writes the remaining data to the file
    output.close(); // Finishes the file
}
```

String portName = Serial.list()[1];	port série COM13
ellipse(posX, val, 10, 10);	Trace un cercle.
output = createWriter("positions.txt");	Crée un fichier texte
output.println(val);	Ecrit la valeur reçue dans un fichier texte
output.close();	Fermeture du fichier

Le fichier texte :

14	28
15	39
16	57
17	71
18	71
19	93
20	111
21	131
22	144
23	155
24	165
25	166

Le résultat graphique avec processing:



Vérification avec un tableur

