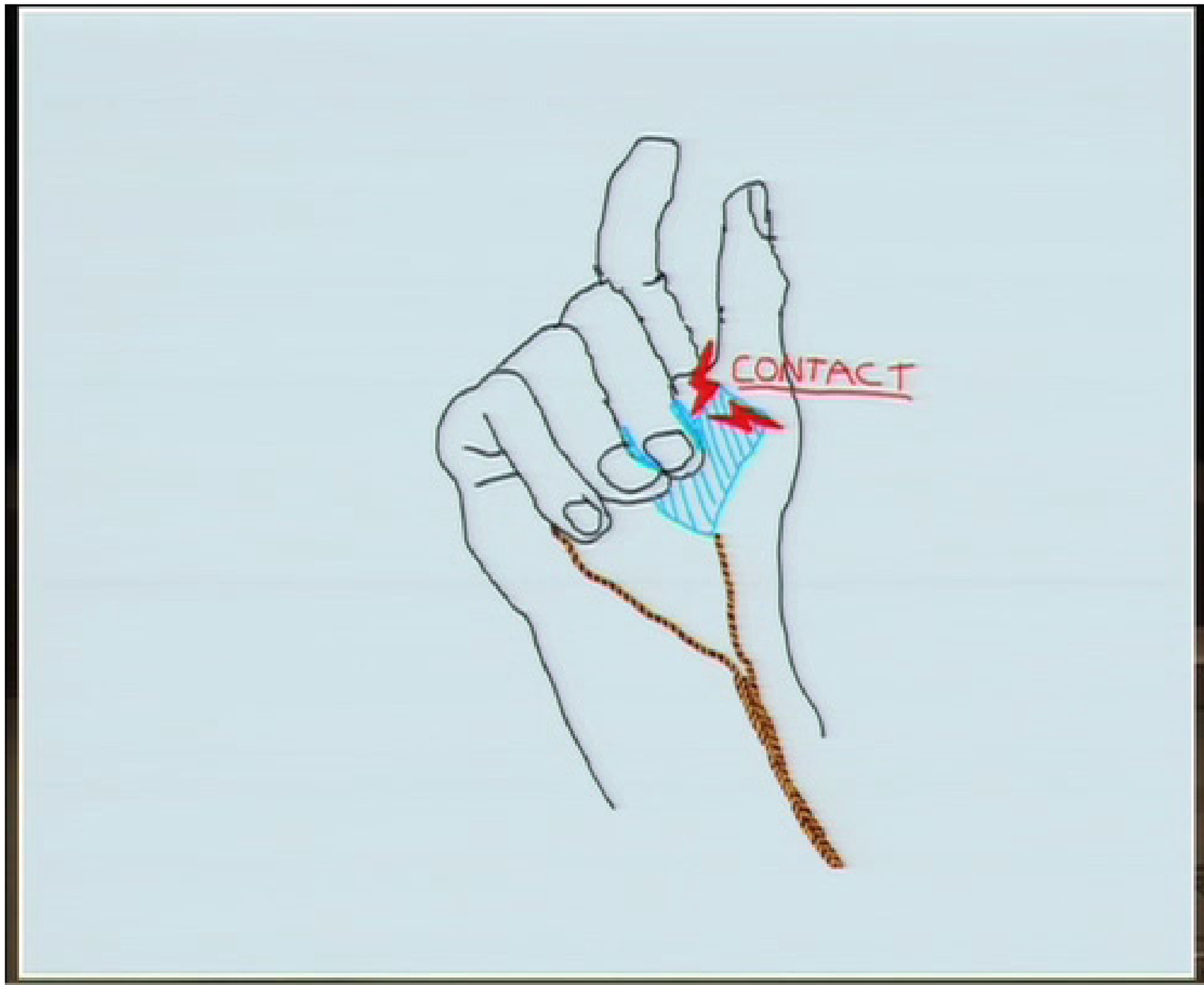
ARDUINO : INPUT



Input

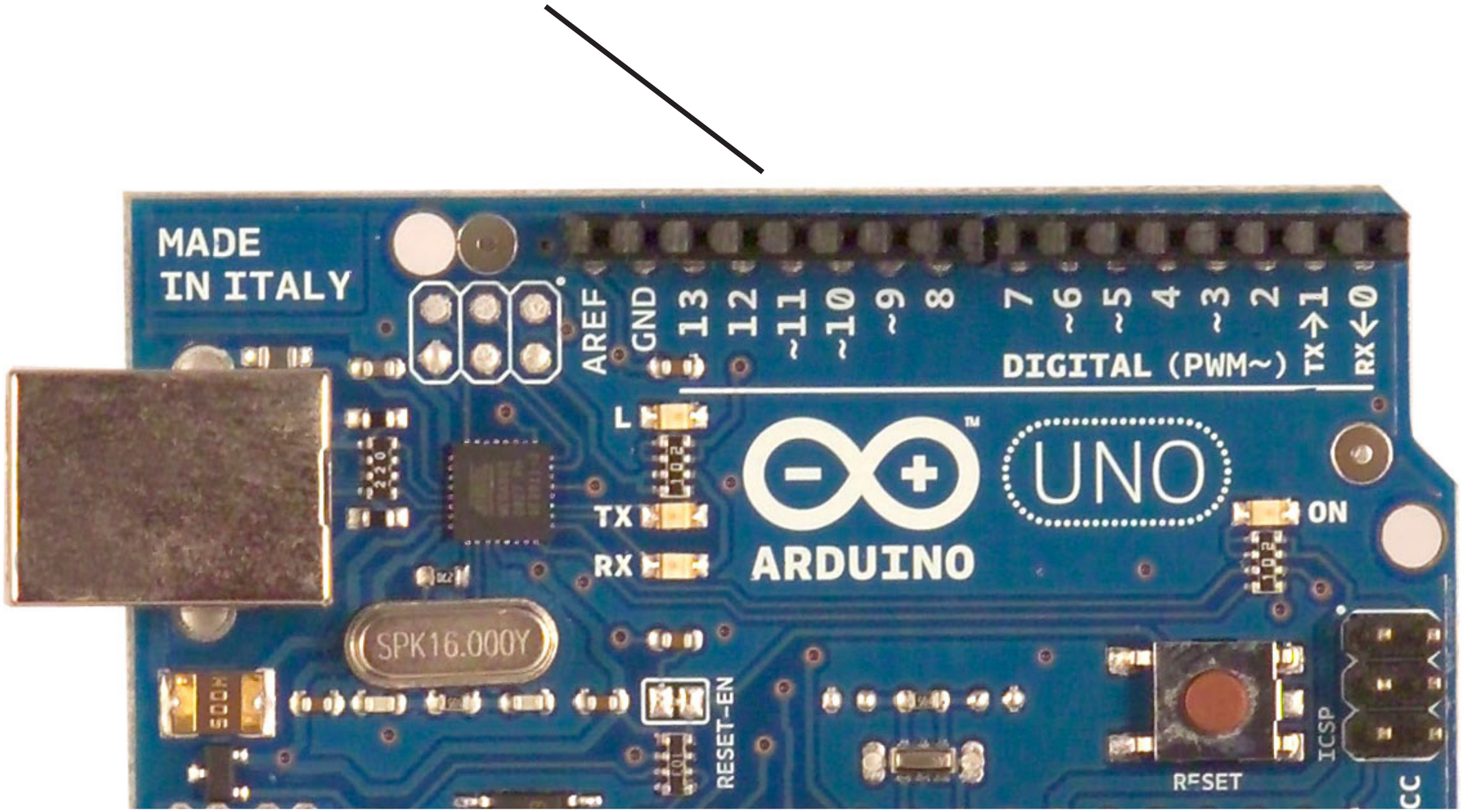
Nous allons d’abord nous intéresser à ce qu’on peut faire entrer de Arduino.

Arduino va capter depuis le monde extérieur des signaux électriques.
Ces signaux peuvent être :
- digitaux
- structurés (digitaux mais sous forme de communication)
- analogiques

Les montages vont être légèrement plus complexes que l’output que nous avons vu, mais pas plus.

les entrées digitales

Des entrées /sorties digitales



Entrées digitales = 1 ou 0

Pour utiliser pour une entrée digitale, il faut juste la déclarer comme telle :

```
void setup(){
digitalmode(8,INPUT);
}
```

une fois que ceci est fait, l'entré «écoute», et elle s'attend à recevoir une tension électrique.

0 volts = 0 ou false ou non
5 volts = 1 ou true ou oui

3 volts = oui
1 volt = non

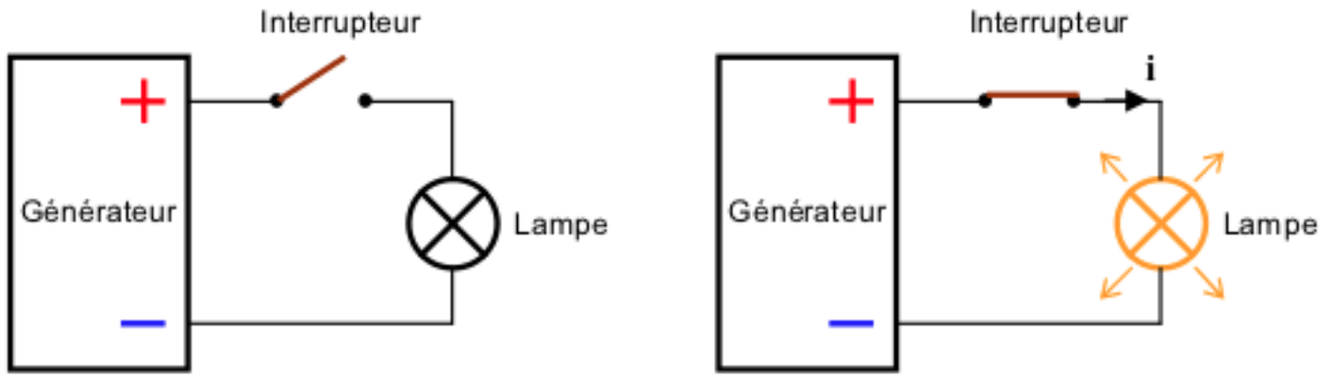
Le digital a besoin de messages clairs.

Cas simple : deux fils

Pour fabriquer un bouton, deux fils suffisent.

Le but est de «fermer le circuit»

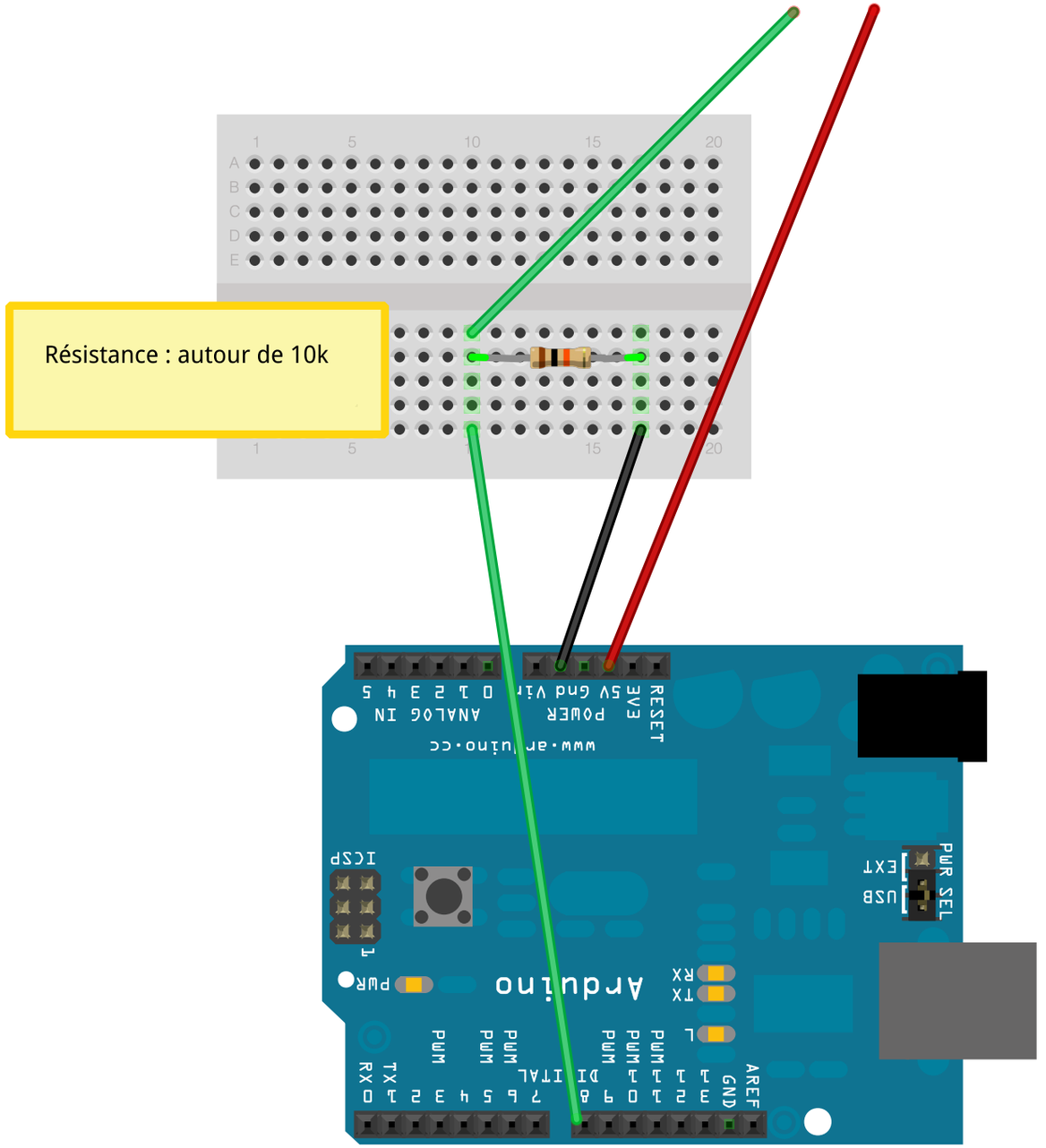
Que la tension sorte de arduino, passe par un cable et revienne vers le capteur.



Montage et programmation

Il y a toujours deux mondes dans un projet arduino : le montage physique et la programmation.

Ici nous allons commencer par le montage physique :



La programmation

Côté software, on va faire ceci :

```
void setup(){
pinMode(8,INPUT);
pinMode(13,OUTPUT);
}

void loop(){
int etat=digitalRead(8);
      if(etat==HIGH){
            digitalWrite(13,HIGH);
      } else {
            digitalWrite(13,LOW);
      }
}
```

OU

```
void loop(){
int etat=digitalRead(8);
digitalWrite(13,etat);
}
```

Pense le bouton

Un bouton est un moyen de fermer un contact. Faire revenir la tension dans arduino.

Dès lors, beaucoup de choses peuvent être un bouton : un bouton récupéré ou «bindé» dans un montage existant, deux bouts de bois, du papier «alu», une fourchette et couteau, etc.

Arduino l’est pas une pile

Le montage que l’on vient de réaliser peut être fait sans arduino...
On appuie et ça s’allume...

Arduino c’est aussi de la programmation. On va donc complexifier le script pour faire clignoter la led à une vitesse tout le temps plus lente.

Une led clignotante, vitesse variable

```
int attente=0;

void setup(){
pinMode(8,INPUT);
pinMode(13,OUTPUT);
}

void loop(){
int etat=digitalRead(8);
      if(etat==HIGH){
            digitalWrite(13,HIGH);
            delay(attente);
            digitalWrite(13,LOW);
            delay(attente);
            attente++;
          } else {
            attente=0;
          }
      }
```

Ensuite, vous pouvez modifier le script :
- la même chose avec un son
- clignoter du plus lent au plus rapide
- faire revenir à 0 progressivement quand on lâche le bouton

Voir ce qu'on fait : Serial.print()

Pour contrôler ce qui se passe à l'intérieur de arduino, on allume ici une led, mais si on fait des choses plus complexes et que ça ne parche pas comme attendu, on doit pouvoir demander à arduino «qu'est-ce que tu fous ?».

Pour cela, on va établir une communication avec lui pour qu'il envoie des messages (on pourra aussi lui envoyer des messages).

On établit en fait une communication de type «série» (comme sur les vieux ordis).

Dans le setup, on ouvre la communication comme ceci:

```
Serial.begin(9600); // on oublie pas la majuscule !
```

et ensuite on peut à tout moment envoyer des messages vers la console via la commande Serial.print() ou Serial.println();

```
Serial.println(«bouton appuyé»);
```

Une led clignotante + série

```
int attente=0;

void setup(){
pinMode(8,INPUT);
pinMode(13,OUTPUT);
Serial.begin(9600);
}
```

```
void loop(){
int etat=digitalRead(8);
      if(etat==HIGH){
            Serial.println(«bouton on !»);
            digitalWrite(13,HIGH);
            delay(attente);
            digitalWrite(13,LOW);
            delay(attente);
            attente++;
          } else {
            attente=0;
            Serial.println(«bouton off !»);
          }
      }
```

Ensuite, vous pouvez modifier le script :

- la même chose avec un son

- clignoter du plus lent au plus rapide

- faire revenir à 0 progressivement quand on lâche le bouton

Envoyer des infos ralentit !

Attention, envoyer des infos par le port série demande des ressources à Arduino et surtout du temps (même si à notre échelle, c'est très rapide).

On évite donc de passer trop de messages, et on retire tout les tests dans un script final, pour de meilleures performances.

Debounce !

Le problème des boutons, c'est qu'ils peuvent avoir de mauvais contacts. Dans ce cas, lorsque le bouton est poussé, pendant une fraction de seconde, on peut avoir une série de contacts on et off très rapide, ce qui peut être un problème si on veut déclencher un événement précis et unique.

On emploiera alors un peu de programmation supplémentaire pour faire ce qu'on appelle un «debounce»

Voir le script d'exemple digital/Debounce

Compter le temps

**L’exemple de Debounce utilise la fonction millis(), qui renvoie un chiffre :
le nombre de millisecondes (1000eme de seconde) depuis que arduino s’est allumé.**

Ce chiffre est rapidement gigantesque, c’est pourquoi il est stocké dans une variable de type “unsigned long“. Comme dans l’exemple suivant :

```
unsigned long temps;
ou
long temps;
```

Pour compter un temps d’action humain, on va prendre un point de référence, puis demander à arduino combien de temps s’est écolé depuis ce point de référence.

Un exemple rapide :

Script pour compter le temps

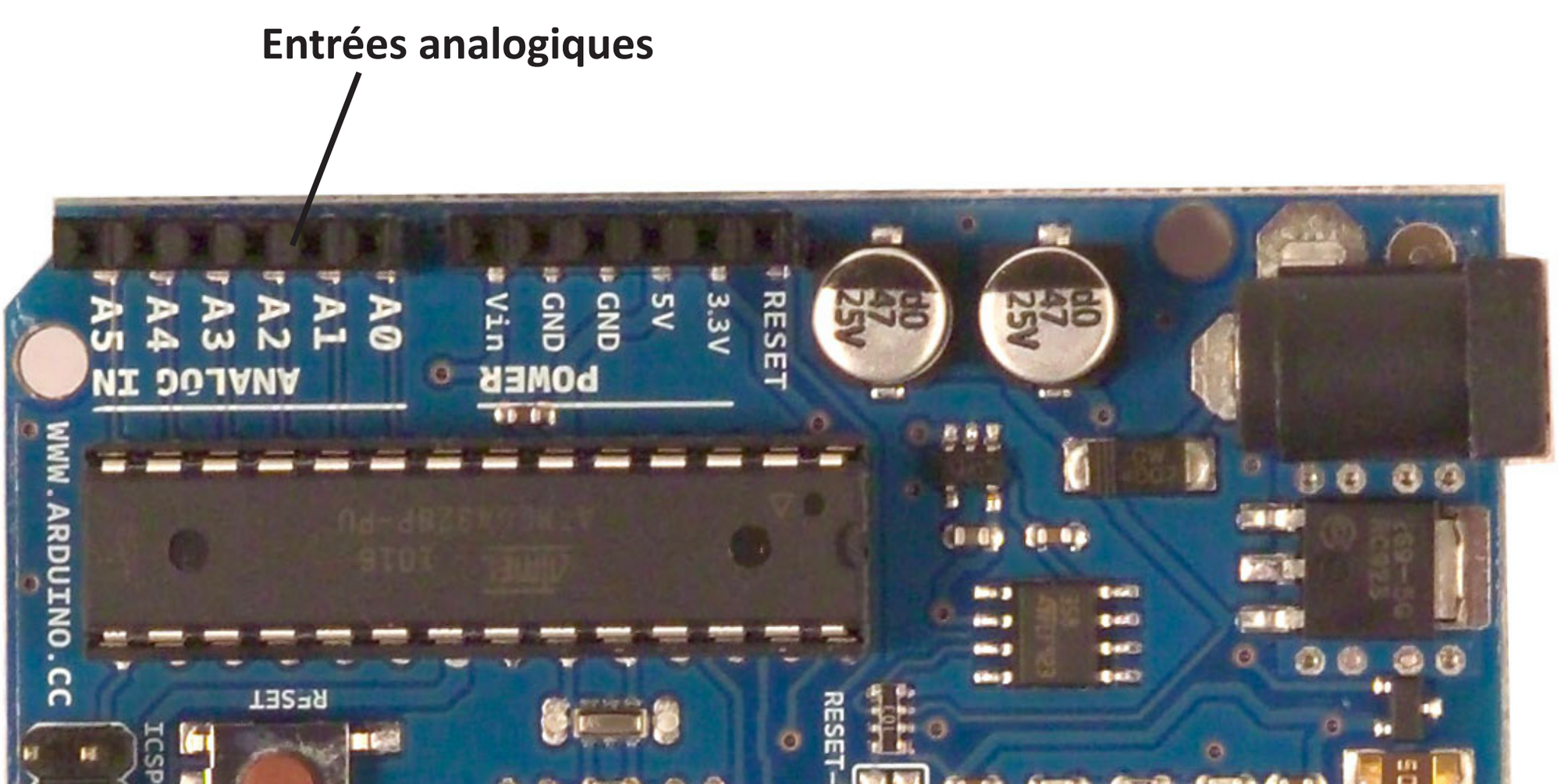
```
int attente=0;
long top;
long duree;

void setup(){
pinMode(8,INPUT);
pinMode(13,OUTPUT);
Serial.begin(9600);
}
```

```
void loop(){
int etat=digitalRead(8);
      if(etat==HIGH){
            if(top==0){
                  top=millis();
            }
            digitalWrite(13,HIGH);
            delay(attente);
            digitalWrite(13,LOW);
            delay(attente);
            attente++;
      } else {
            duree=millis()-top;
            Serial.println(duree);
      }
}
```

Entrée analogique

On peut capter des messages qualitatifs avec arduino. Pas seulement oui ou non, mais une information du genre «un peu» ou «beaucoup».

Ce sont les entrées analogiques qui le permettent.



Analogique = 1024 valeurs

Avec les entrées analogiques, on peut percevoir 1024 paliers dans la réception de la tension électrique.

0 volts= 0
5 volts= 1023

Du coup, tout une série de capteurs vont pouvoir être utilisés :
- Capteur de lumière
- Potentiomètre
- piezo
- ...

et aussi des capteurs complexes
- Gaz
- Mouvement
- etc.

Pas de setup

Contrairement aux entrées digitales, les entrées analogiques ne sont pas déclarées dans le setup. On récupère directement la valeur avec analogRead(pin)

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

void loop(){
    int val=analogRead(0);
    Serial.println(val);
    delay(100); // eviter de surcharger le port serie
}
```

Le potentiomètre

Un potentiomètre est une résistance variable, un composant qui permet de doser la quantité de flux électrique. Les potentiomètres sont identifié par leur résistance en ohms. 10K, 47K par exemple. Il existe des potentiomètres rotatifs de toutes sortes et des sliders (dit «à glissière»).

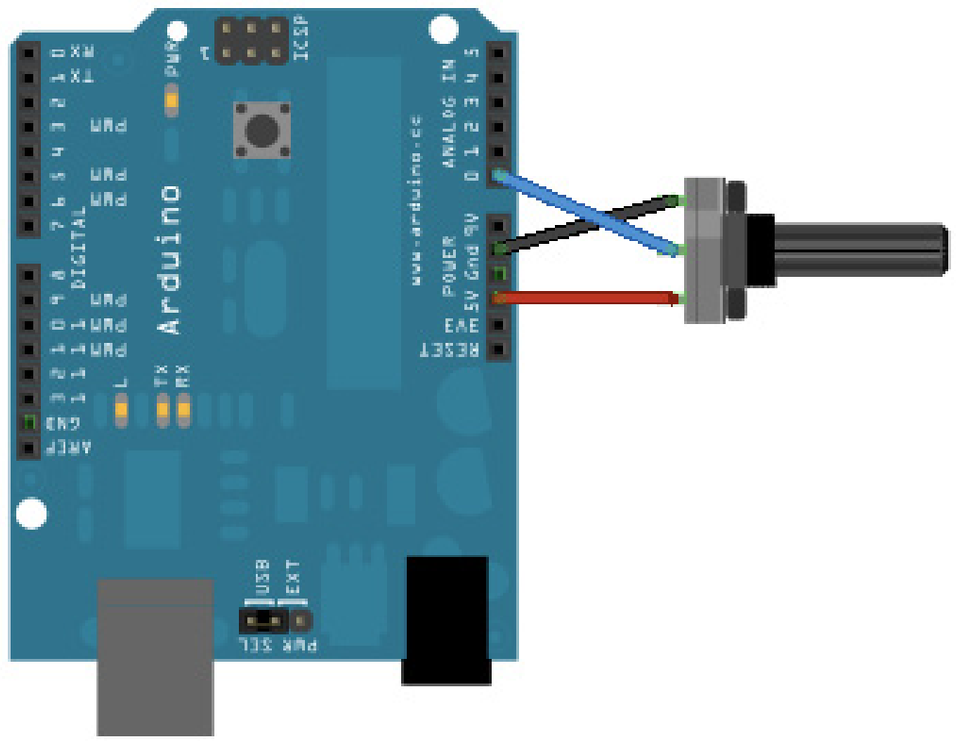
Un potentiomètre se compose de trois sorties :
On placera l'entrée analogique sur la sortie du centre

Le 5 volts d'un côté

Le ground de l'autre.

On peut tester avec le script Analog/AnalogInput

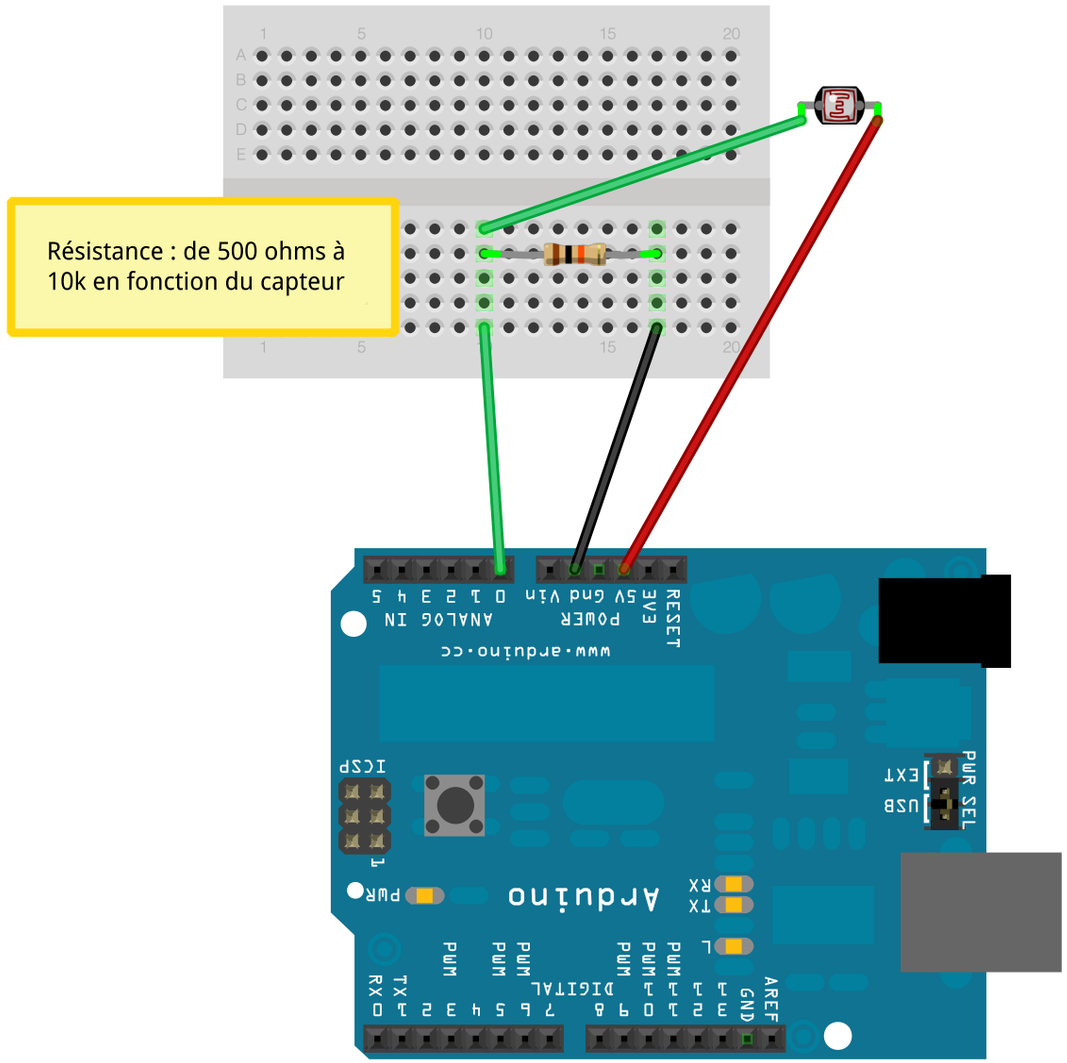
(ajoutez un Serial.print() pour monitorer)



Un LDR

Le LDR (Light Dependent Resistor) est une résistance variable, comme le potentiomètre, sauf qu'ici c'est la lumière reçue sur le capteur qui détermine la puissance du flux électrique qui le traverse. Le capteur est d'une taille variable (de quelques millimètres à un centimètre environ).

Voici un montage simple. Le script peut être le même que pour le potentiomètre.



Let’s roll

En avant.

Fabriquez un prototype avec
une entrée, capture d’un événement par arduino

et une sortie, c’est à dire un événement produit par arduino.