

Arduino

Introduction N°2

Simon CHOLLET



- Introduction
- Hardware : Composants matériels
- Software : le logiciel
- Pratique
-
-
- Conclusion



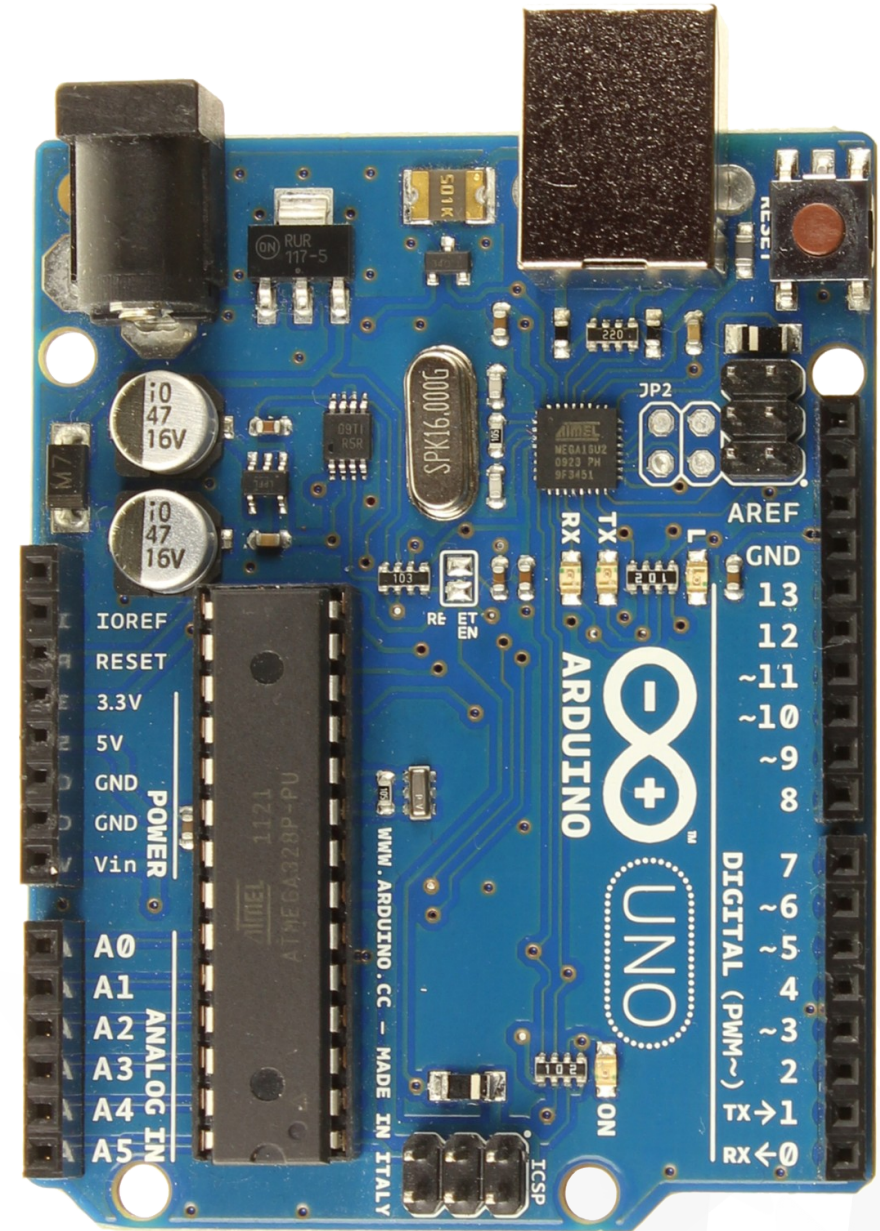
Introduction

- Arduino ... mais qu'est-ce que c'est ?
- Pourquoi utiliser un Arduino ?
- La famille (officielle) !

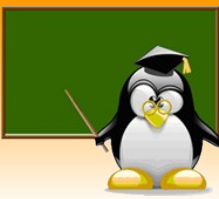


Arduino ... mais qu'est-ce que c'est ?

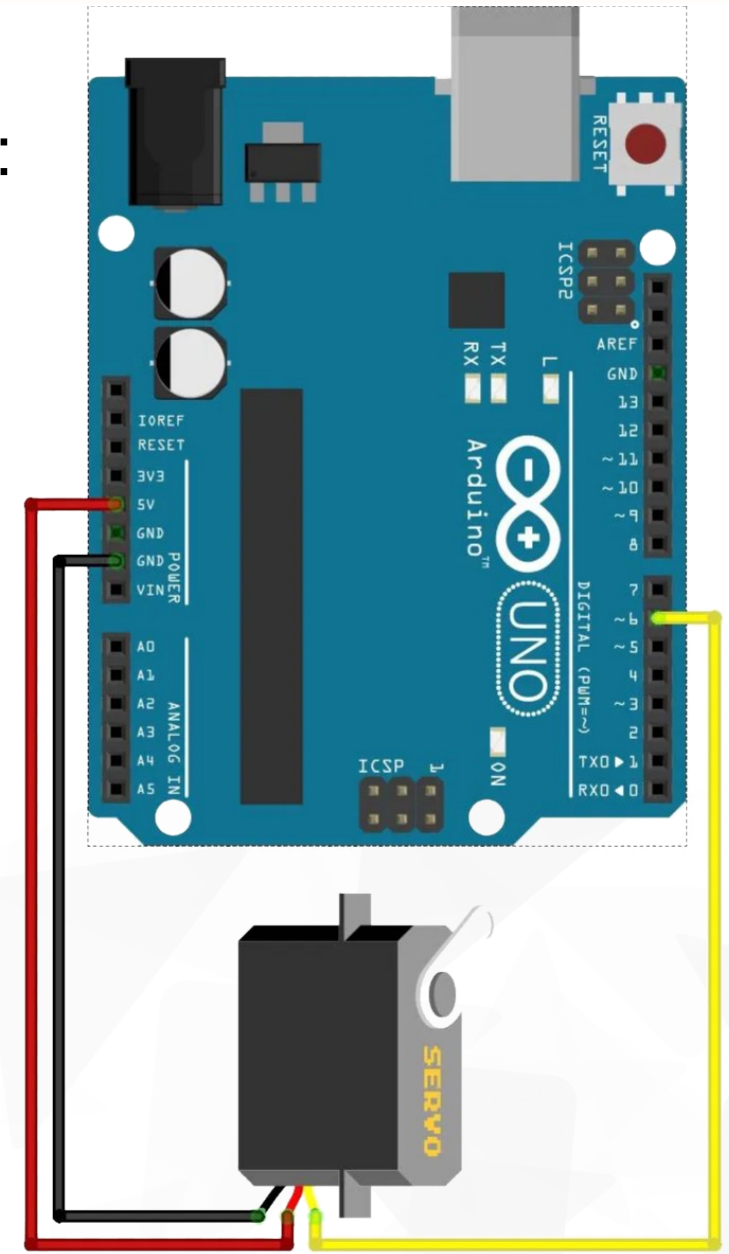
- Platine de **développement** :
 - Connexions avec l'extérieur
 - Circuit électronique avec **micro-contrôleur**
- **Open source** : tous les schémas sont disponibles
- Basée sur du **matériel** et des **logiciels** « faciles » à utiliser
- **Matériel** : capteurs, moteurs, éclairages, etc.
- **Logiciel** : éditeur de code (écriture programme), programmes déjà faits



Pourquoi utiliser Arduino ?



- **Débuter** et **apprendre** à programmer
- **Bricoler** des petits objets électroniques : robots, éclairages, etc.
- **S'initier** à l'électronique, s'**amuser** !
- **Peu** coûteux
- Possible d'acheter des **clones** encore moins chers
- Communauté **énorme**, beaucoup :
 - **Tutoriels** : vidéos, pages Internet
 - **Forums** de discussion
- Complètement **Open Source** :
 - on sait ce qu'il y a dedans





La famille (officielle) !



Arduino Uno



Arduino Leonardo



Arduino Robot



Arduino Esplora



Arduino Micro



Arduino Pro Mini



Arduino Due



Arduino Yún



Arduino BT



Arduino Mega 2560



Arduino Pro



Arduino Fio



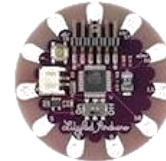
Arduino Mega ADK



Arduino Ethernet



LilyPad Arduino USB



LilyPad Arduino Simple



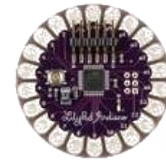
Arduino Mini



Arduino Nano



LilyPad Arduino SimpleSnap



LilyPad Arduino





Hardware : Composants matériels

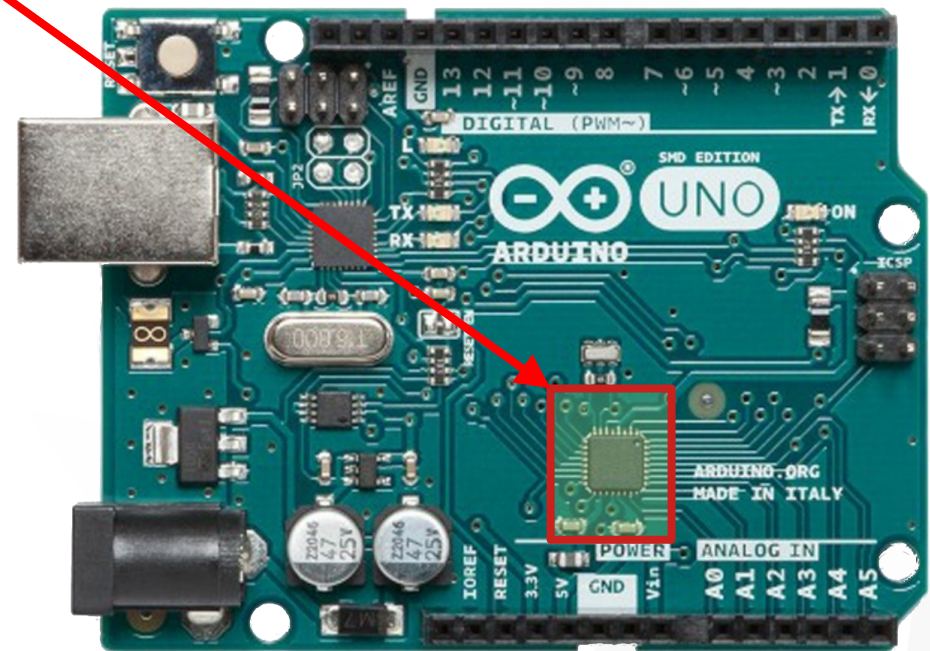
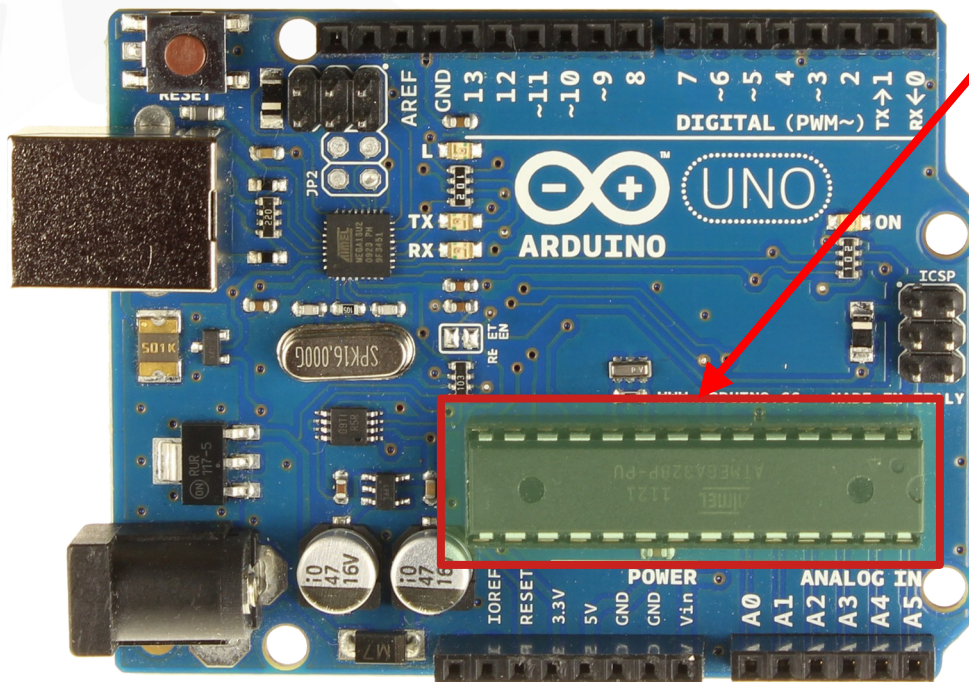
- Le microcontrôleur
- Connectique
- « Pinout » complet
- Plaque d'expérimentation
- Servomoteur



Le microcontrôleur

- **Cerveau** de la carte
- **Stocke** dans sa mémoire et **exécute** le **programme**
- Il peut alors faire : clignoter une LED, afficher des caractères sur un écran, envoyer des données à un ordinateur, mettre en route ou arrêter un moteur ...

Microcontrôleur





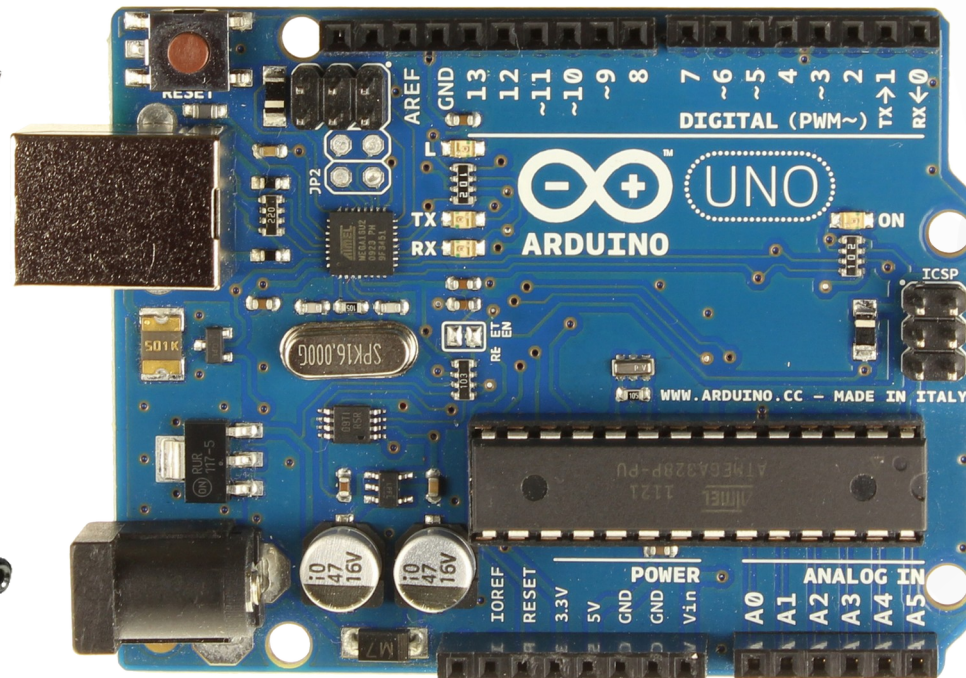
Alimentation

- Pour fonctionner : besoin d'**alimentation**
- Possibilités :
 - En **5V** par le port **USB**
 - Alimentation **externe** (comprise entre 7V et 12V)
 - → **Régulateur** réduit la tension à 5V

USB

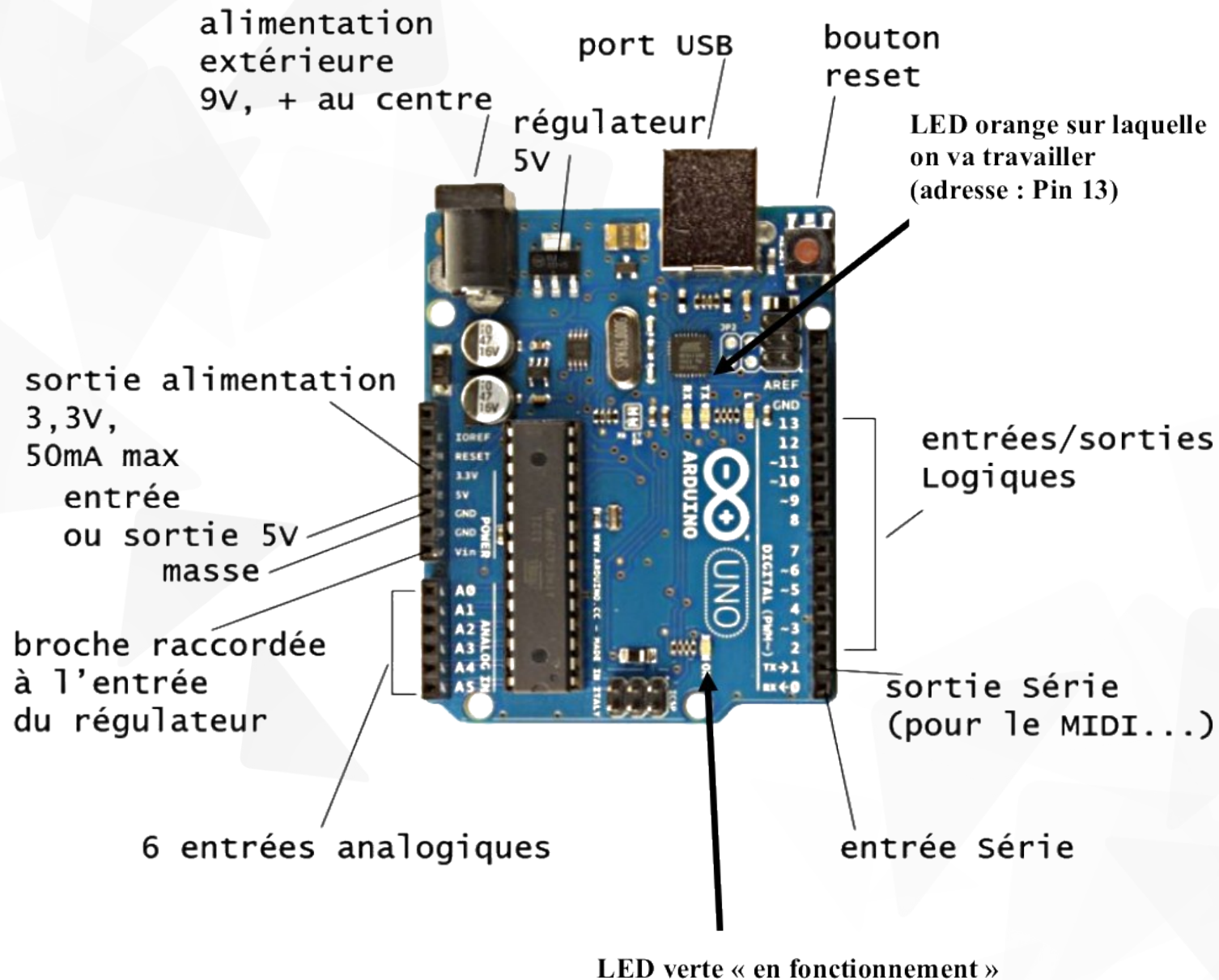


Externe





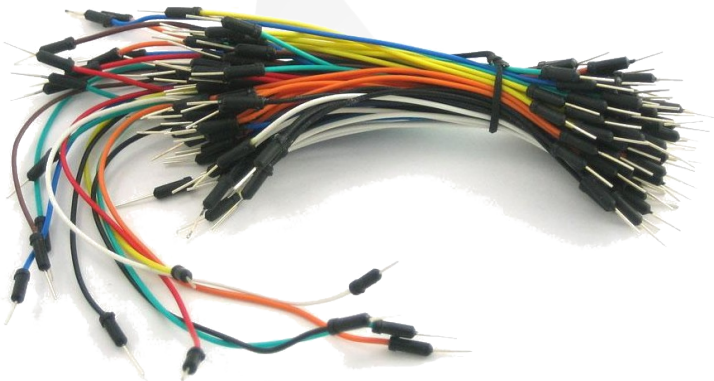
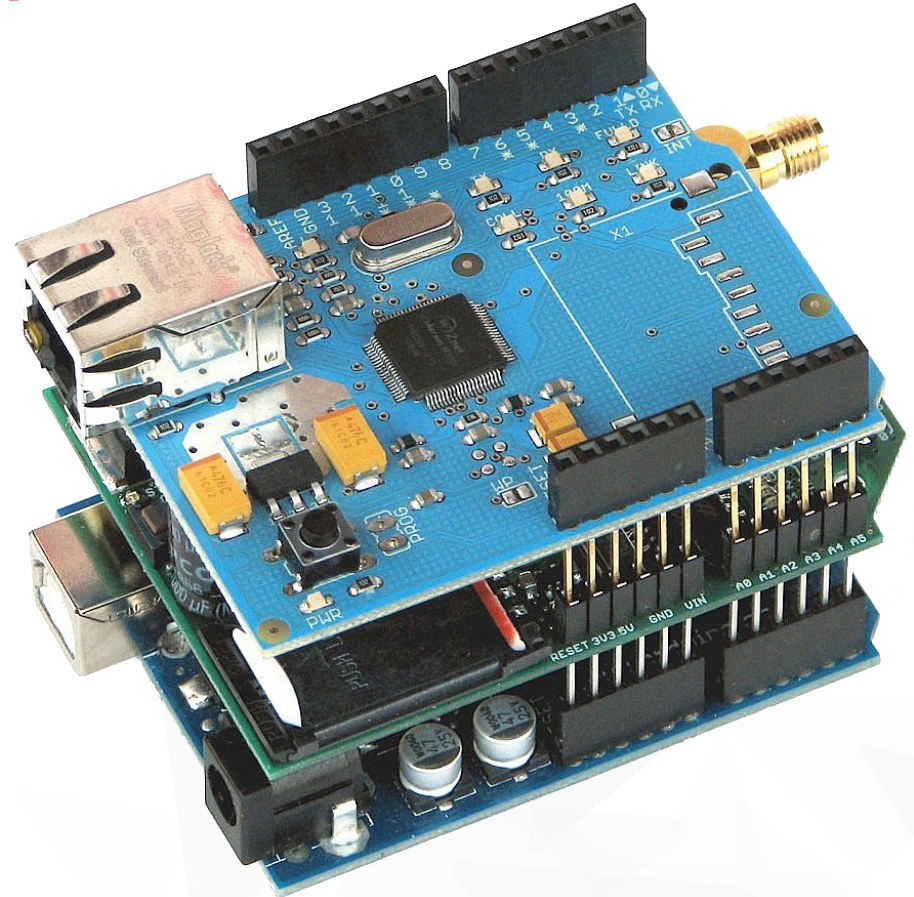
Connectique





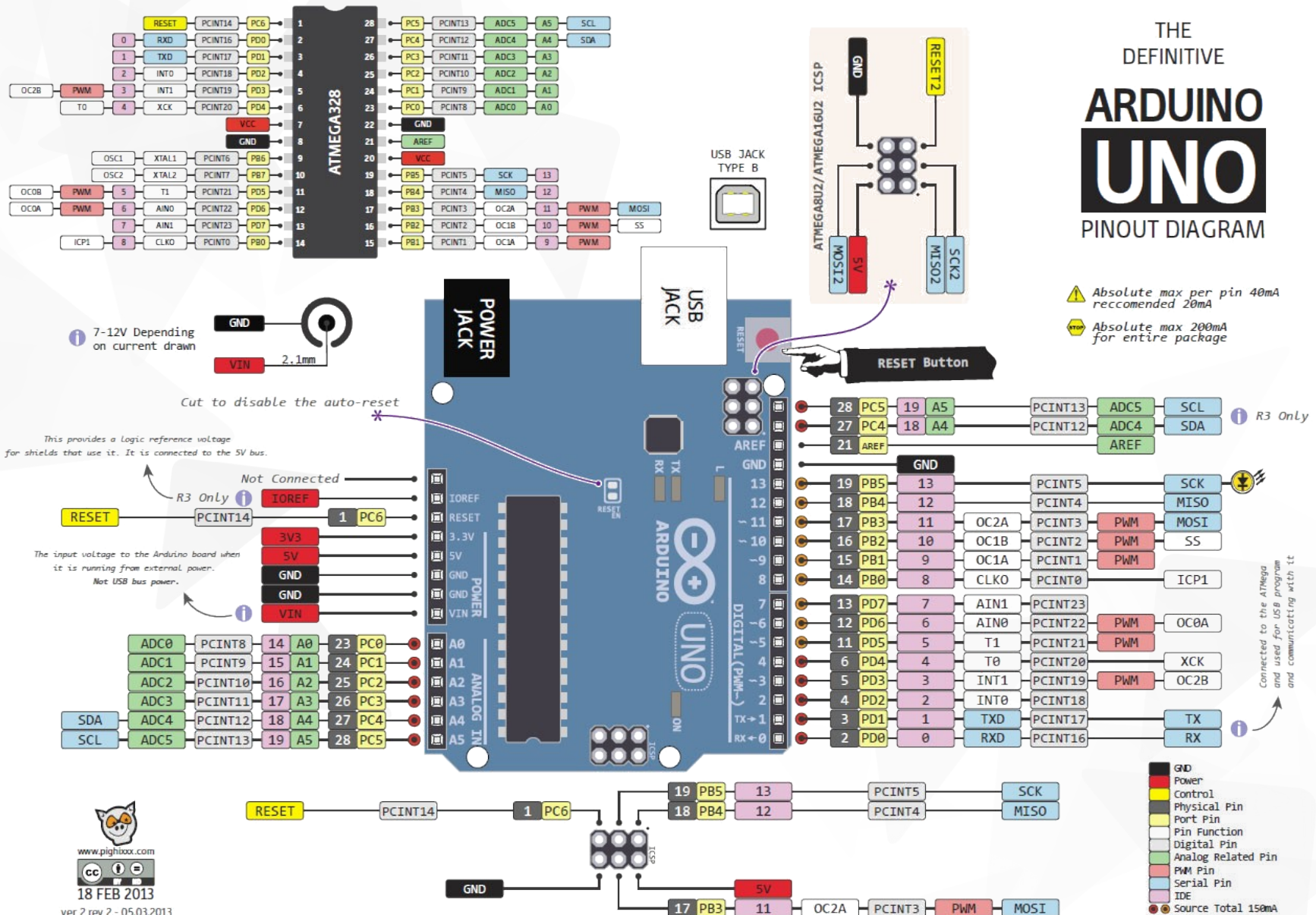
Connectique

- **0 à 13** : Entrées/sorties **numériques**
- **A0 à A5** : Entrées/sorties **analogiques**
- **GND** : Masse (0V)
- **5V** : Alimentation 5V
- **3.3V** : Alimentation 3.3V
- **Vin** : Alimentation non stabilisée
(= même que entrée carte)





« Pinout » complet



www.pighixxx.com
18 FEB 2013
ver 2 rev 2 - 05.03.2013





Plaque experimentation

- Permet de réaliser des montages **sans soudure**
- Toutes sur le **même** principe de connexions :

Longues rangées utilisées pour alimentation : masse (GND, en bleu) et +5V (rouge)

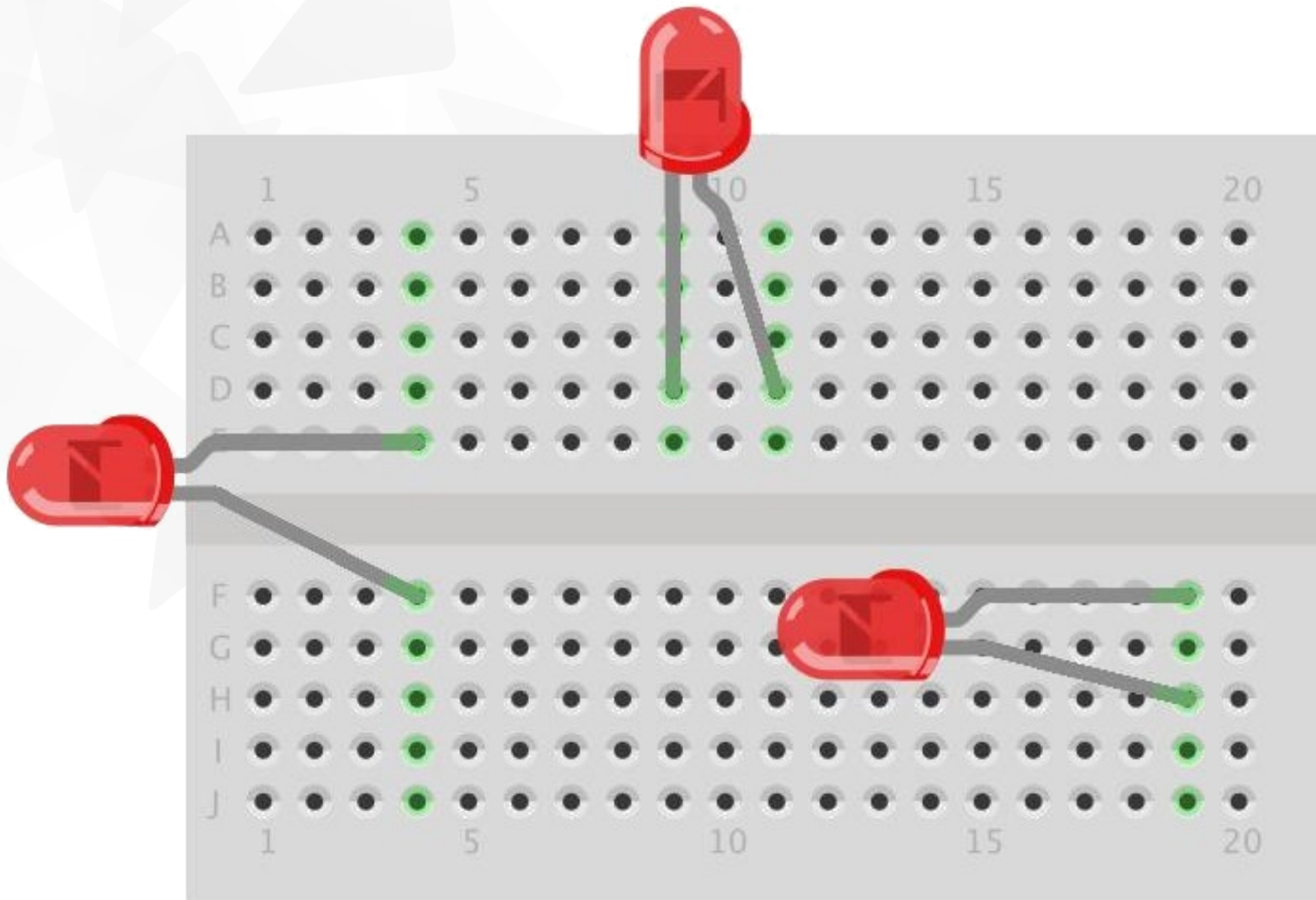
Connecteurs électriquement reliés entre-eux par 5

Connecteurs ligne extérieure sont électriquement reliés entre-eux





Quiz ... Qu'est-ce qui est correct ?



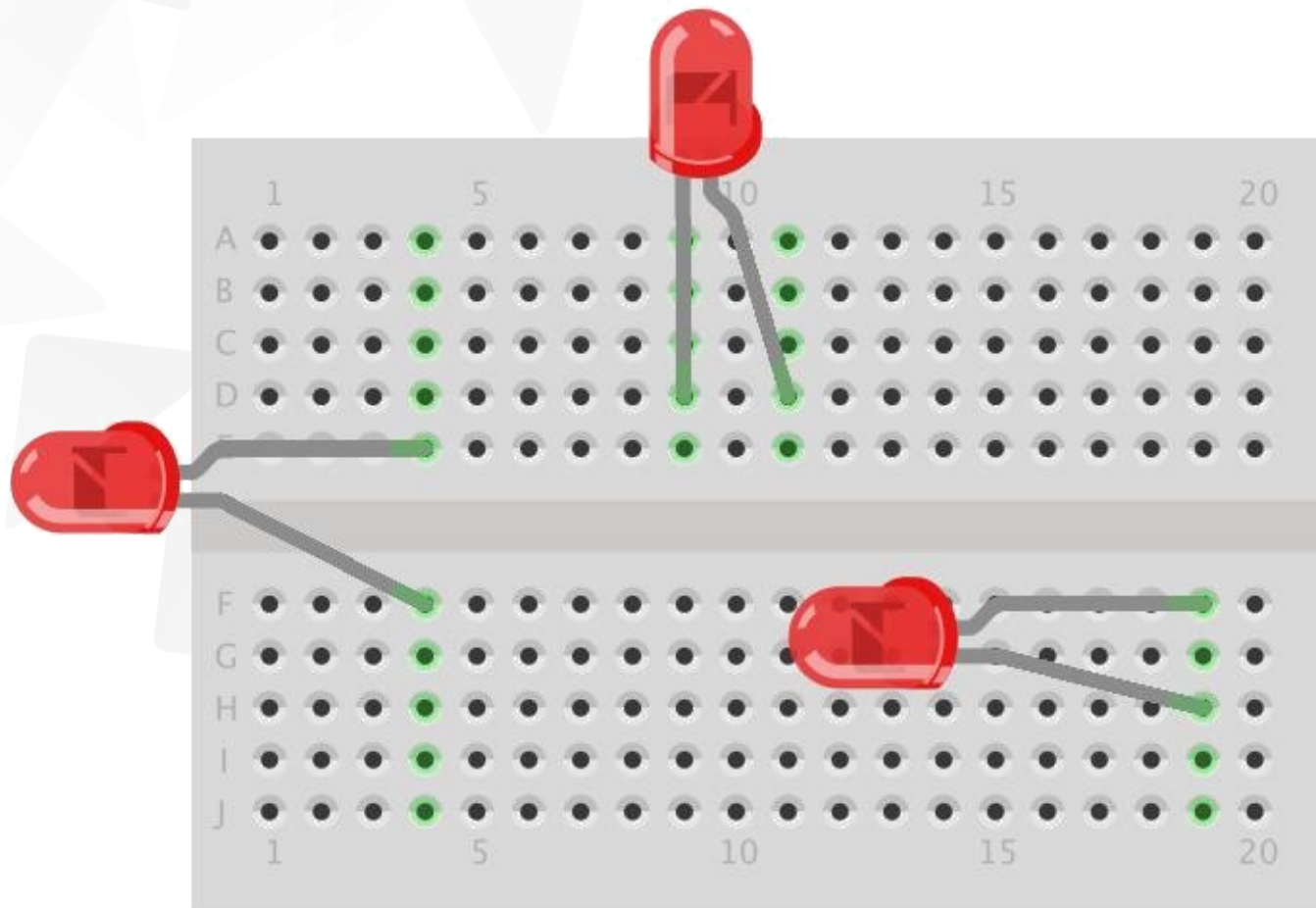


Quiz : réponses !

Juste

Juste

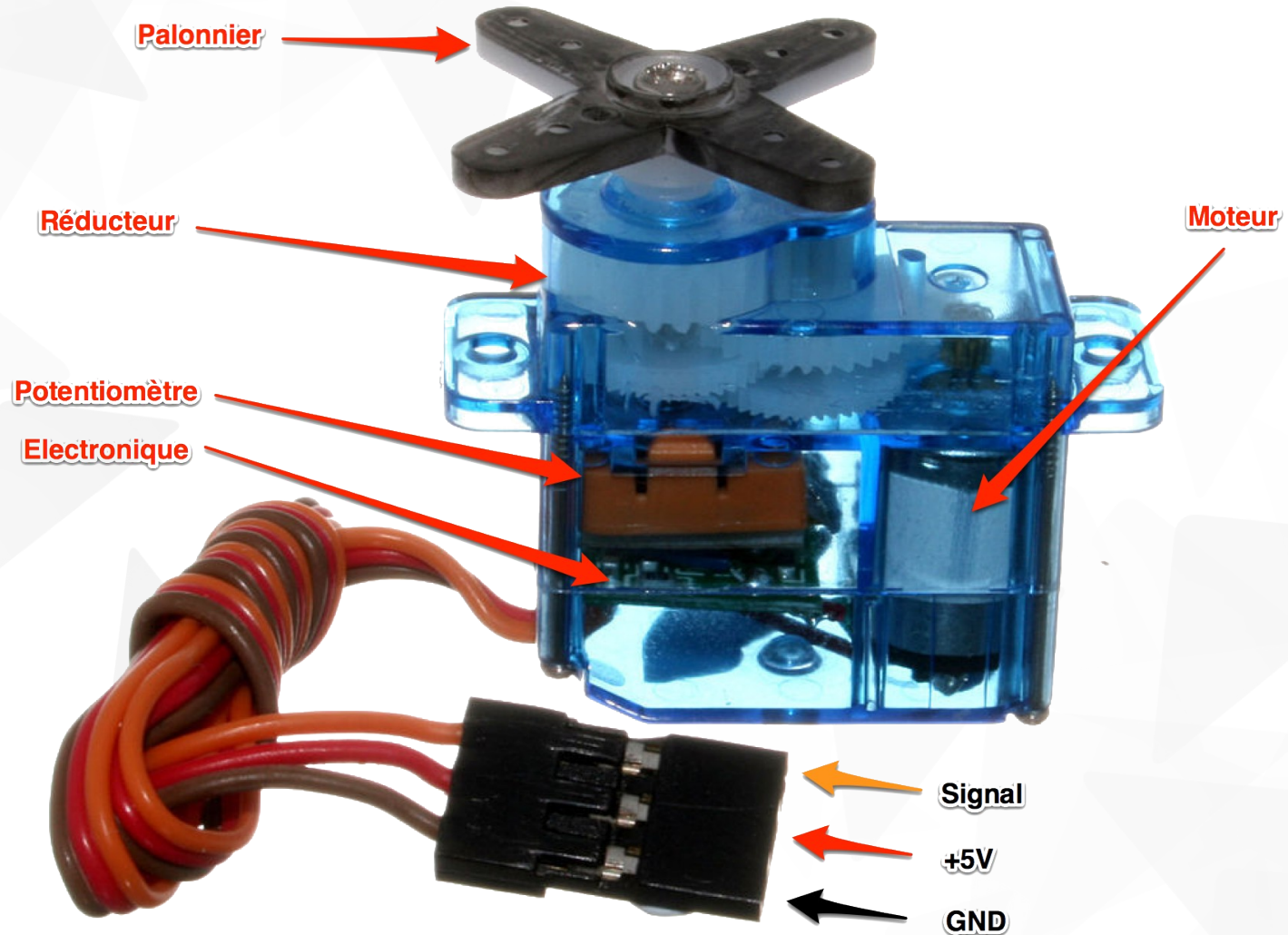
Faux





Servomoteur (1/2)

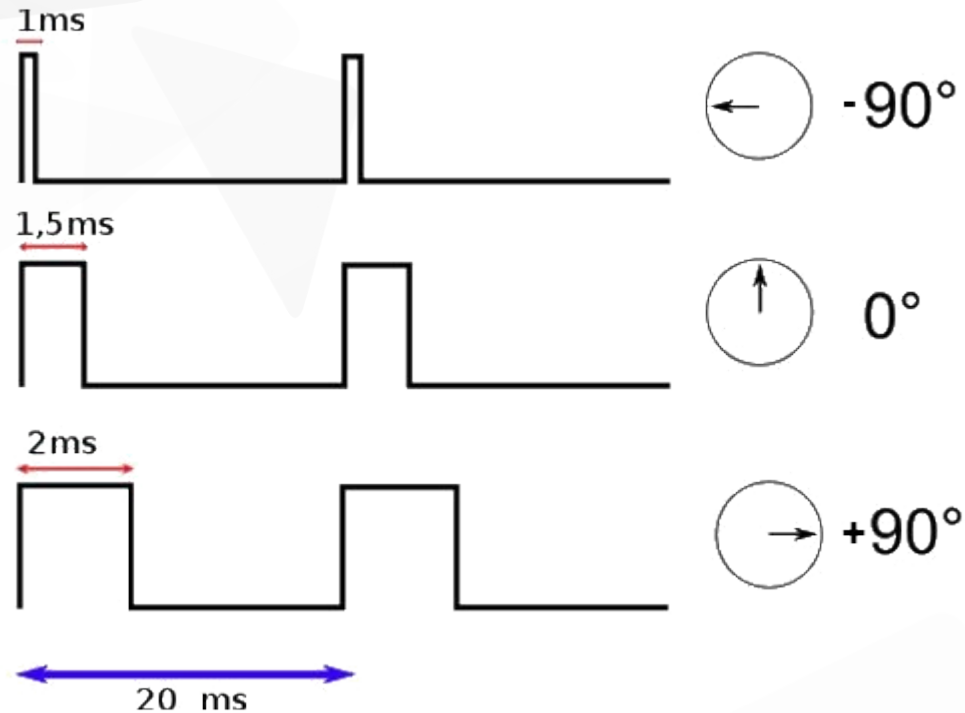
- But : faire **tourner** des choses





Servomoteur (2/2)

- Pilotage par **variation** de **largeur** d'impulsion :



- Une **bibliothèque** de fonctions est incluse !





Software : le logiciel

- Arduino IDE
- En local, à distance
- Les zones
- Le menu



Arduino IDE

- Fonctionne sur : Mac, Windows et Linux
- **Créer**, **tester** et **envoyer** les programmes sur l'Arduino
- Éditeur assez **rudimentaire** ...
- Permet de mettre des **couleurs** sur la **syntaxe**, dans le programme : **coloration syntaxique**
- Téléchargeable : <http://arduino.cc>, pour l'utiliser sur son ordinateur
- Possible de l'utiliser **en ligne** (connecté à Internet) : <https://create.arduino.cc/editor>



Arduino IDE : en local



```
TP2 | Arduino 1.6.11

digitalWrite(p_rouge, HIGH); //feu piéton du rouge
delay(3000); //durée 3 secondes

buttonState = digitalRead(buttonPin); //lecture de l'état du bouton
if ((buttonState != memoire) && (buttonState == HIGH)) //Comparaison de l'état du bouton pc
//si l'état du bouton est différent
//stocké dans "mémoire" (=LOW), alors
{
    digitalWrite(verte, LOW); //feu vert éteint
    digitalWrite(orange, HIGH); //feu orange allumé
    delay(1000); //durée 1 seconde
    digitalWrite(orange, LOW); //feu orange éteint
    digitalWrite(rouge, HIGH); //feu rouge allumé
    digitalWrite(p_vert, HIGH); //feu piéton vert allumé
    digitalWrite(p_rouge, LOW); //feu piéton rouge éteint
    delay(5000); //durée 5 secondes
    digitalWrite(p_vert, LOW); //feu piéton vert éteint
}
else //sinon on exécute le code suivant, soit le cycle des feux par
{
    digitalWrite(verte, LOW); //feu vert éteint
    digitalWrite(orange, HIGH); //feu orange allumé
    delay(1000); //durée 1 seconde
    digitalWrite(orange, LOW); //feu orange éteint
    digitalWrite(rouge, HIGH); //feu rouge allumé
    delay(3000); //durée 3 secondes
}

5 Arduino/Genuino Uno sur /dev/tty.usbmodem1421
```



Arduino IDE : en ligne





Arduino IDE : les zones





Arduino IDE : Le menu



- Bouton **1** : **Vérifier** le programme (recherche d'erreurs)
- Bouton **2** : **Envoyer** du programme sur l'Arduino
- Bouton **3** : **Créer** un nouveau fichier
- Bouton **4** : **Ouvrir** un fichier existant
- Bouton **5** : **Enregistrer** un fichier
- Bouton **6** : **Console** série (pour l'affichage)
- Un programme **ouvert** = une **nouvelle** fenêtre



Arduino IDE : Fenêtre de programme



Prog1_blink

Nom du fichier

```
/*
Code 1 - Programmation de l'Arduino
Objectif: faire clignoter la LED montée sur la broche 13
*/

//***** FONCTION SETUP = Code d'initialisation *****
// La fonction setup() est exécutée en premier et une seule fois, au démarrage du programme
void setup()
{
  // Initialise la broche 13 comme sortie
  pinMode(13, OUTPUT);
}

//***** FONCTION LOOP = Boucle sans fin = coeur du programme *****
// la fonction loop() s'exécute sans fin en boucle aussi longtemps que l'Arduino est sous tension
void loop()
{
  // Met la broche 13 au niveau haut = allume la LED
  digitalWrite(13, HIGH);

  // Pause de 1000ms = 1 seconde
  delay(1000);

  // Met la broche 13 au niveau bas = éteint la LED
  digitalWrite(13, LOW);

  // Pause de 1000ms = 1 seconde
  delay(1000);
}
```

Programme





Arduino IDE : Fenêtre de contrôle

Enregistrement terminé.

30 ESP32 Dev Module, Disabled, Default 4MB with spiiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi, BT, QIO, 80MHz, 4MB (32Mb), 921600, None sur /dev/ttyUSB0

Messages de l'application

Zone d'affichage erreurs

ESP32 Dev Module, Disabled,

None sur /dev/ttyUSB0

État de la connexion avec l'Arduino



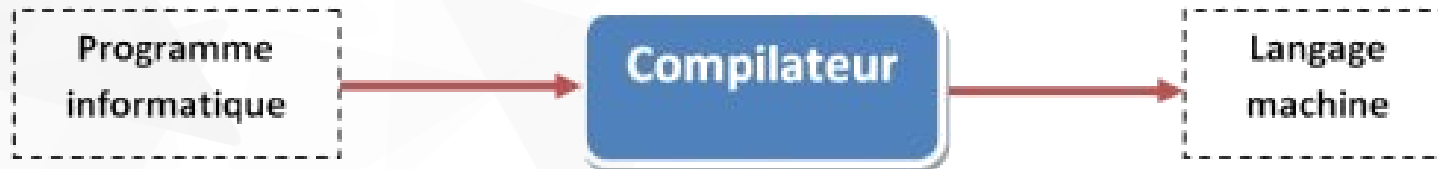


Software : la programmation

- Compilation
- La syntaxe du langage
- Déroulement du programme
- Quelques éléments de syntaxe
- Variables – Constantes
- Les types de variables
- Conditions
- Boucles



Compilation



Traduction



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

```
"00010011001110001110101001110
00111111011010001000000100011
10001000010000111111100111100
11001100010000011100011100101
00011011101010101000100111000
01100010000111111001111000101
11001001111110011111011001110
00110101101001010001100111000
110101101011010011101111101100
01010101011011011011110111010
01011011010111011101011011111"
```





La syntaxe du langage

- **Syntaxe** = règles d'écriture du langage
- Langage très proche du **C/C++**
- Le code **minimal** :

```
// Fonction d'initialisation de la carte  
void setup()  
{  
    // Contenu de l'initialisation  
}
```

```
// Fonction principale, elle se répète (s'exécute) à l'infini  
void loop()  
{  
    // Contenu de votre programme  
}
```

- **Pas obligé** de remplir les 2 fonctions
- Mais **obligé** de les écrire

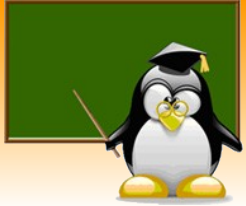




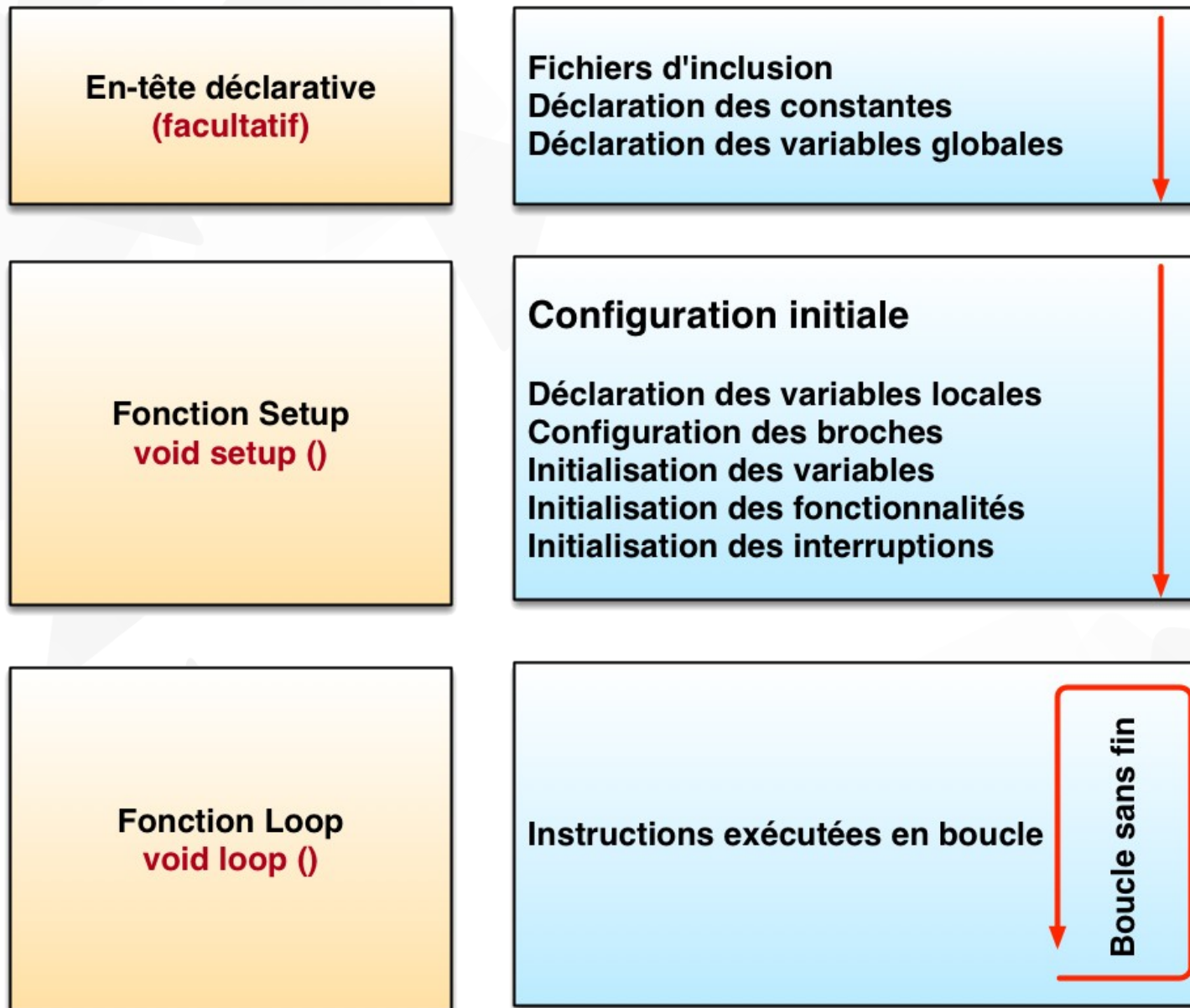
Quelques règles

- **Attention** aux majuscules, minuscules
- **Pas** de chiffre au début des noms
- Pas d'**accent**, sauf dans les commentaires
- Lisibilité du code :
 - Mettre des **lignes vides** pour aérer
 - Indenter le code : tabulations
- Nom de variable compréhensible
-





Déroulement du programme





Quelques éléments de syntaxe

- **Instructions** : lignes de code qui disent au programme : ‘fait ceci, fait cela, ...’
- ‘;’ : terminent les instructions
- ‘{’ et ‘}’ : ‘**conteneur**’, utilisées pour le **contenu** de :
 - Une fonction
 - Une condition
 - Une boucle
- **Commentaires** : lignes de code **non exécutées**
 - Pour **une** ligne : ‘//’
 - Pour **plusieurs** lignes : ‘/* */’
 - **Exemples** :

```
// Cette ligne est un commentaire sur UNE SEULE ligne
/* Cette ligne est un commentaire, sur PLUSIEURS lignes
   qui sera ignoré par le programme, mais pas par celui qui lit le
   code */
```

W
X
A
T
Z
Y
S





Variables – Constantes

- **Variables** :
 - Valeurs qui **peuvent changer**
 - Peuvent avoir une **valeur initiale**
 - Doivent être définies avec un **type**
 - Exemples :

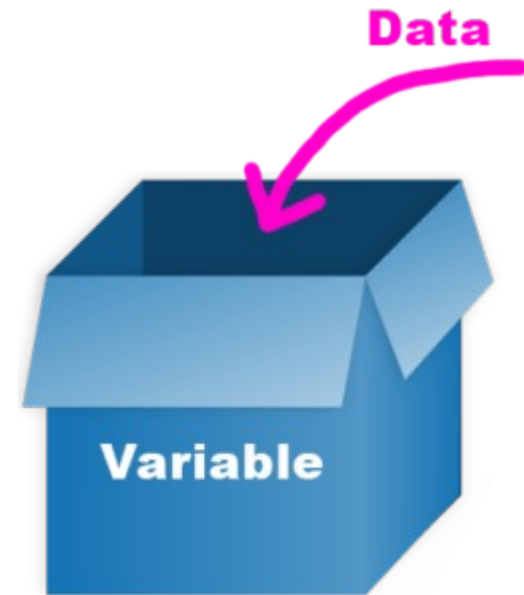
```
int compteur = 0;
```

```
float temperature = 21.5;
```

- **Constantes** :
 - Valeurs qui **ne changent pas**
 - Définies par la syntaxe : **#define**
 - Souvent en **majuscules**
 - Exemples :

```
#define LED_ROUGE = 13;
```

```
#define TEMP_ATTENTE = 1.2;
```





Les types de variables

- Types **signés** => **négatifs** ou **positifs**

Type	Min .. Max	Nb. bits	Nb. Octets
int	-32 768 à +32 767	16 bits	2 octets
long	-2 147 483 648 à +2 147 483 647	32 bits	4 octets
char	-128 à +127	8 bits	1 octet
float	-3.4×10^{38} à $+3.4 \times 10^{38}$	32 bits	4 octets
double	-3.4×10^{38} à $+3.4 \times 10^{38}$	32 bits	4 octets

- Types **non signés** => **positifs** (unsigned)

Type	Min .. Max	Nb. bits	Nb. Octets
unsigned char	0 à 255	8 bits	1 octet
unsigned int	0 à 65 535	16 bits	2 octets
unsigned long	0 à 4 294 967 295	32 bits	4 octets

- Types **supplémentaires**

Type	Min .. Max	Nb. bits	Nb. Octets
byte	0 à 255	8 bits	1 octet
word	0 à 65 535	16 bits	2 octets
boolean	0 à 1	1 bit	1 octet

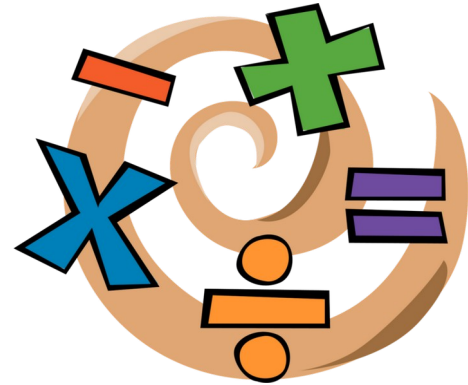




Opérations – Maths

- Opérations **simples** :

- Addition : **+** → **A = 12 + 3;** // A = 15
- Soustraction : **-** → **A = 10 - 4;** // A = 6
- Multiplication : ***** → **A = 10 * 2;** // A = 20
- Division : **/** → **A = 10 / 5;** // A = 2
- Modulo (reste division entière) : **%** → A = 5 % 2; // A = 1



- Combinaisons** des opérateurs possible :

- Incrémentation : **+=** → **A += 2;** // c-a-d : A = A + 2
- Décrémentation : **-=** → **A -= 2;** // c-a-d : A = A - 2
- Multiplication : ***=** → **A *= 2;** // c-a-d : A = A * 2
- Division : **/=** → **A /= 2;** // c-a-d : A = A / 2
- Possible d'**autres** : **sin()**, **tan()**, **exp()**, **sqrt()**, etc.





Conditions

- Choix à faire entre **plusieurs propositions**
- **Tester** les variables
- Opérateurs de **comparaison** :

Symbole	Signification
==	A == B : A est égal à B ?
<	A < B : A est inférieur à B ?
>	A > B : A est supérieur à B ?
<=	A <= B : A est inférieur ou égal à B ?
>=	A >= B : A est supérieur ou égal à B ?
!=	A != B : A est différent de B ?

- Opérateurs **logiques** :

Symbole	Signification
&&	CondA && CondB : Condition A et condition B sont vraies
 	CondA CondB : Condition A ou condition B est vraie
!	CondA ! CondB : Condition A est différente de condition B





Conditions

```
if (condition1) {  
    action1();  
}  
else {  
    action2();  
}
```

```
if (condition1) {  
    action1();  
}  
else if (condition2) {  
    action2();  
}  
else {  
    action3();  
}
```

```
switch (valeur) {  
    case 0:  
        action1();  
        break;  
  
    case 1:  
        action2();  
        break;  
  
    default:  
        action3();  
        break;  
}
```





Boucles

- Boucle **While** : tant que condition est **vraie**, alors 'action'

```
while(condition) {  
    action  
}
```

- Exemple :

```
int compteur = 0 ;  
while(compteur < 5) {  
    compteur++ ;  
}
```

- Boucle **For** : exécuter un **nombre de fois** (nb) 'action'

```
for(i=0 ; i < nb ; i++) {  
    action  
}
```

- Exemple :

```
for(int i=0 ; i< 5 ; i++) {  
    print(i);  
}
```





Software : la programmation

- Fonctions sur entrées/sorties digitales
- Fonctions sur entrées/sorties analogiques
- Fonctions sur lien série



Fonctions sur entrées/sorties digitales

- Fonction **pinMode(noBroche, mode)** :
 - Configure broche → entrée ou sortie
 - Exemple : configuration broche 13 en sortie :
pinMode(13, OUTPUT);
- Fonction **digitalWrite(noBroche, valeur)** :
 - Ecrit un état (haut = **HIGH** ou bas = **LOW**) sur broche
 - Exemple : met à l'état haut la broche 13 :
digitalWrite(brocheLED, HIGH);
- Fonction **digitalRead(noBroche)** :
 - Lit l'état (haut ou bas) d'une broche
 - Exemple : lit l'état de la broche 13 :
etat = digitalRead(13);



Fonctions sur entrées/sorties analogiques



- Fonction **analogRead(noBroche)** :
 - Lit la valeur analogique sur une broche
 - Exemple : lit la valeur sur la broche A3 :
valeur = analogRead(3);
- Fonction **analogWrite(noBroche, valeur)** :
 - Écrit une valeur analogique (un PWM = modulation en largeur d'impulsion) sur une broche
 - Exemple : écrit une valeur analogique sur le PWM1 de 250 :
analogWrite (9, 250);





Fonctions sur lien série (1/2)

- Fonction **Serial.begin(debit)** :
 - Fixe la vitesse de communication avec le lien série
 - Exemple : règle la vitesse à 9600 bps :
Serial.begin(9600);
- Fonction **Serial.available()** :
 - Retourne le nombre de caractères disponibles sur le lien série
 - Exemple : nombre de caractères disponibles :
int nbCarac = Serial.available();
- Fonction **Serial.read()** :
 - Lit un caractère disponible sur le lien série
 - Exemple : lit un caractère :
char carac = Serial.read();





Fonctions sur lien série (2/2)

- Fonction **`Serial.print(donnees {, type })`** :
 - Envoie des données sur le port série.
 - Exemple : envoie le message 'Bonjour !' :
`Serial.print("Bonjour !");`
- Fonction **`Serial.println({donnees {, type}})`** :
 - Envoie des données sur le port série, avec saut de ligne
 - Exemple: envoie le message 'Bonjour !', puis 'A tout le monde' :
`Serial.println("Bonjour !");`
`Serial.println("A tout le monde");`





Pratique

- Clignotement LED
- Chenillard avec 4 LED



Clignotement LED

- Faire un **nouveau fichier** → clic sur bouton :
- **Recopier** le code ci-dessous :



```
Prog1_blink 5
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);

  delay(1000);

  digitalWrite(13, LOW);

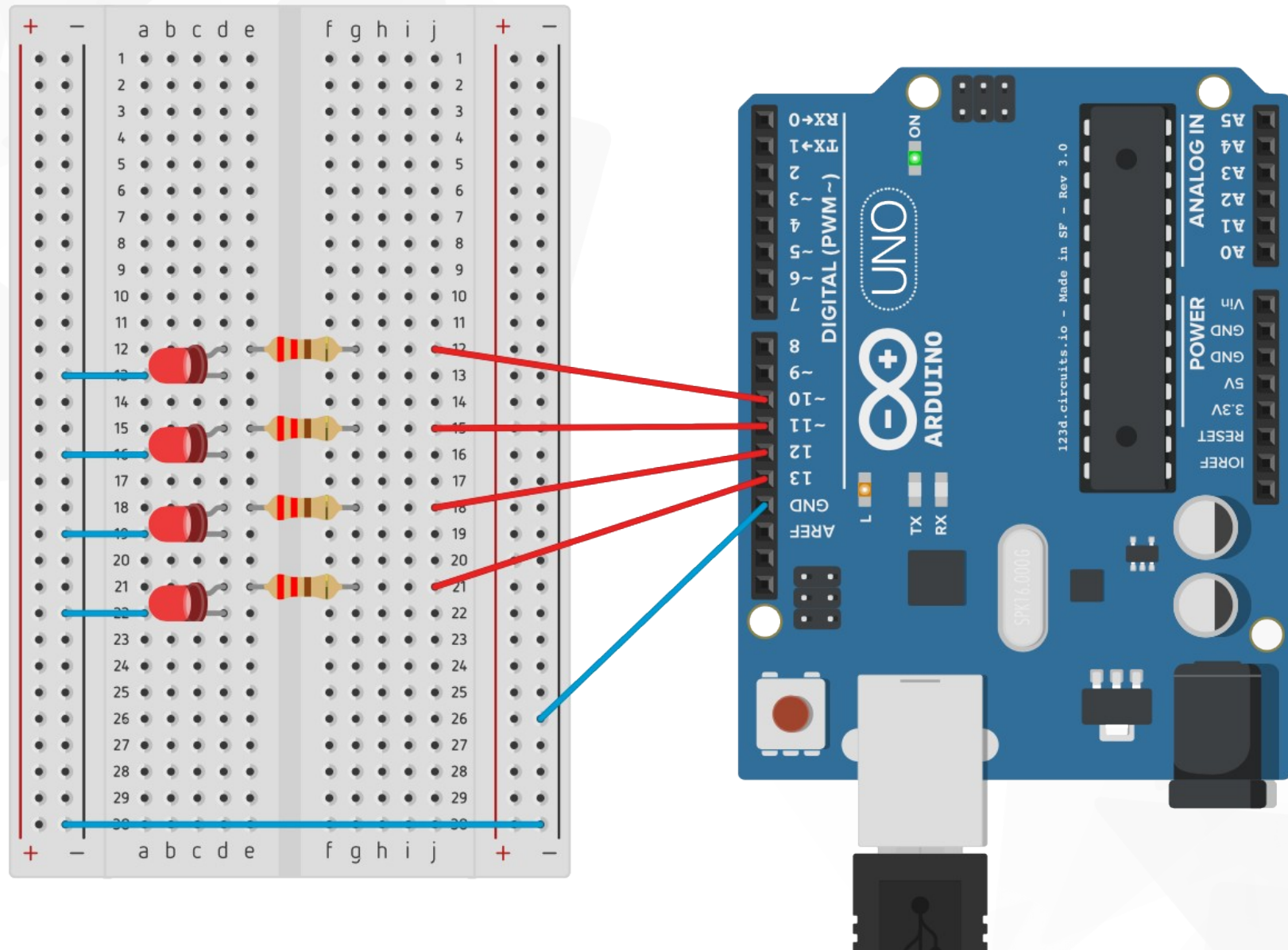
  delay(1000);
}
```

- **Envoyer** vers l'Arduino → clic sur bouton :
- **Bonus** : changer '**1000**' par : '**500**' ou '**2000**'





Chenillard 4 LED : montage





Chenillard 4 LED : programme

- Faire un **nouveau fichier** → clic sur bouton :
- **Recopier** le code ci-dessous :



```
Prog2_chenillard4 §
int i;

void setup()
{
  for(i=0; i < 4; i++) {
    pinMode(10 + i, OUTPUT);
  }
}

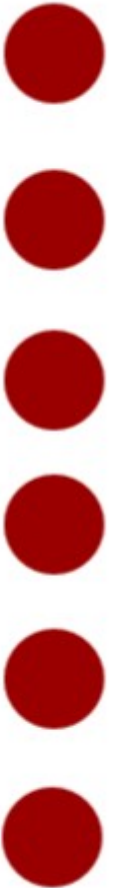
void loop()
{
  for(i=0; i < 4; i++) {
    digitalWrite(10 + i, HIGH);

    delay(100);

    digitalWrite(10 + i, LOW);

    delay(100);
  }
}
```

- **Envoyer** vers l'Arduino → clic sur bouton :
- **Bonus** : changer '**100**' par : '**500**' ou '**1000**'





Servomoteur

- Faire un **nouveau fichier** → clic sur bouton : 
- **Faire** : Croquis → Inclure une bibliothèque → Servo
- **Redémarrer** le logiciel
- **Recopier** le code ci-dessous (et l'envoyer) :

```
#include <Servo.h>
```

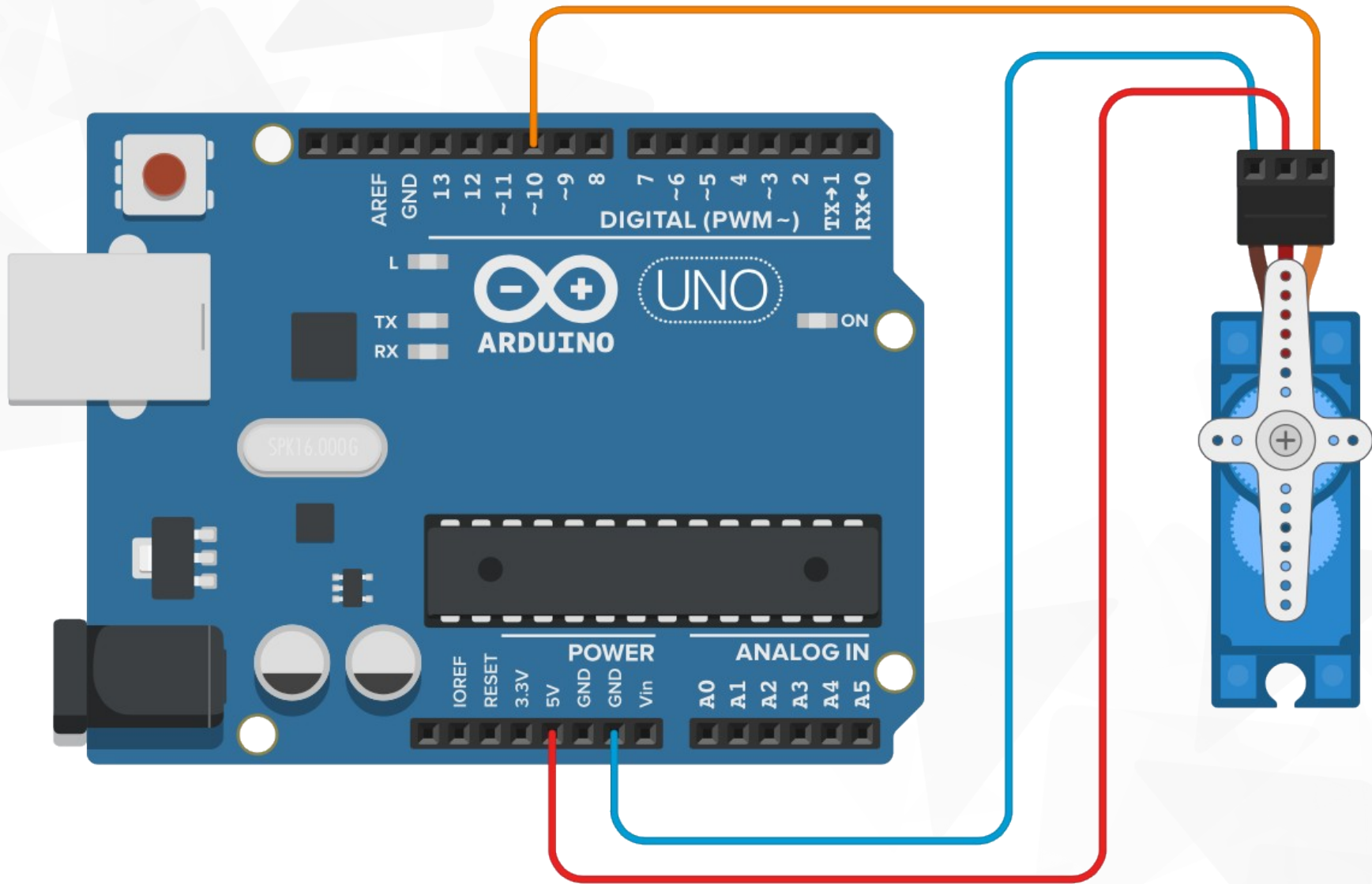
```
Servo monServo;
```

```
void setup()  
{  
  monServo.attach(9);  
}
```

```
void loop()  
{  
  for (int pos=0; pos<=180; pos++) {  
    monServo.write(pos);  
    delay(15);  
  }  
}
```



Servomoteur : montage





Servomoteur 2

- Faire un **nouveau fichier** → clic sur bouton : 
- **Faire** : Croquis → Inclure une bibliothèque → Servo
- **Redémarrer** le logiciel
- **Recopier** le code ci-dessous (et l'envoyer) :

```
#include <Servo.h>
```

```
Servo monServo;
```

```
int pinmonServo=9;
```

```
int pinPotar=A0;
```

```
void setup()
```

```
{
```

```
  monServo.attach(pinmonServo);
```

```
}
```

```
void loop()
```

```
{
```

```
  int valeurPotar=analogRead(pinPotar);
```

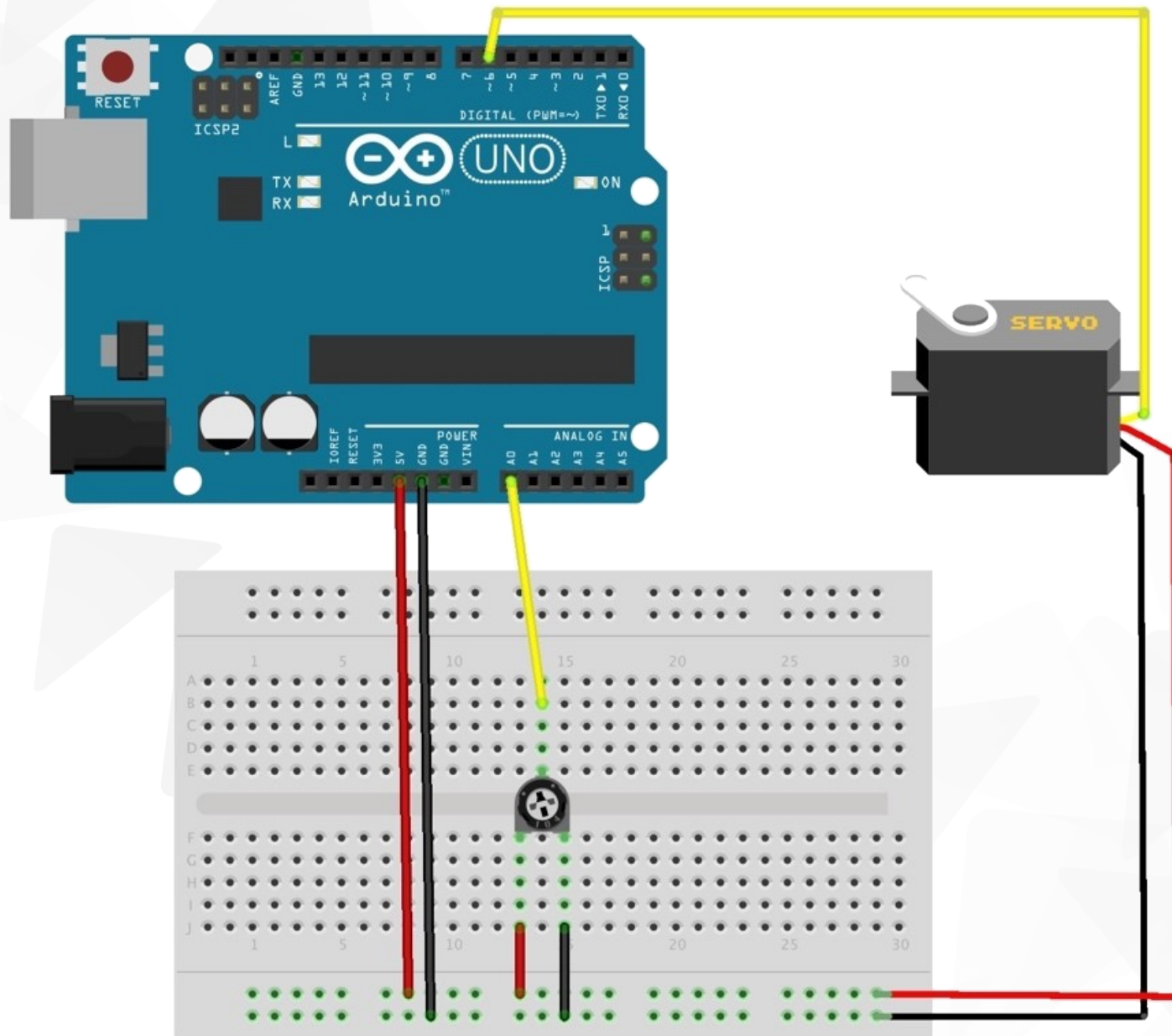
```
  int angle=map(valeurPotar, 0,1023,0,180);
```

```
  monServo.write(angle);
```

```
}
```



Servomoteur 2 : montage





Conclusion

- Récapitulatifs rapides
- Ressources Internet
-



Récapitulatif rapide (1/4)

- **Syntaxe** du langage :
 - **;** : point-virgule pour la fin d'une instruction
 - **{ }** : accolades pour bloc d'exécution
 - **//** : commentaire d'une ligne
 - **/* */** : commentaire multi-lignes
 - **#define** : définition de constante
- **Structures** de contrôle :
 - **if () { } .. else { }** : condition
 - **switch () { case : break; default : break; }**
- **Boucles** :
 - **for (i=0; i<fin ; i++) { }** : boucle de i=0 à i=fin
 - **while() { }** : tant que





Récapitulatif rapide (2/4)

- **Variables** :
Éléments utilisés pour stocker valeurs changeantes
- **Constantes** :
Éléments non modifiables par programme
- **Types** :
 - **char** : caractère
 - **int** : entier
 - **float** : réel
- Constantes **prédéfinies** :
 - **HIGH** : état haut
 - **LOW** : état bas
 - **INPUT** : entrée
 - **OUTPUT** : sortie





Récapitulatif rapide (3/4)

- E/S **numériques** :
 - **pinMode(noBroche, mode)** : configuration
 - **digitalWrite(noBroche, valeur)** : écriture
 - **int digitalRead(noBroche)** : lecture
 - **unsigned long pulseIn(noBroche, valeur)** : durée impulsion
- E/S **analogiques** :
 - **int analogRead(noBroche)** : lecture
 - **analogWrite(noBroche, valeur)** : écriture
- Communication **série** :
 - **Serial.begin(debit)** : ouverture
 - **Serial.available()** : test
 - **Serial.read()** : lecture
 - **Serial.print(data), Serial.println(data)** : écriture





Récapitulatif rapide (4/4)

- Gestion du **temps** :
 - **unsigned long millis()** : nb ms depuis début
 - **delay(nbMilliSecondes)** : attente **ms**
 - **delayMicroseconds(nbMicroSecondes)** : attente **µs**
- **Mappage** de valeurs (transformation d'échelle) :
 - **map(valeur, valMin, valMax, echMin, echMax)**





Ressources Internet

- **Outils** disponibles **gratuitement** sur Mac et PC :
 - **Fritzing** : schémas et PCB
 - **Solve Elec** : schémas électriques avec analyses et simulations
 - **Fido Cadj** : schémas et circuits imprimés
 - **QelectroTech** : plutôt des schémas électriques
 - **iCircuit** : pour dessiner et simuler des circuits
 - **EasyEDA** : pour les schémas, mais aussi le design de PCB ...
- **Simulateur** assez complet : **ThinkerCAD** :
 - Conception **3D**
 - Circuits avec **Arduino**
 - **Codeblocks**
 - **Inscription** au site pour utiliser les différents outils



FIN !

Merci ...

- ... pour avoir tenu jusqu'ici,
- ... de ne pas hésiter à poser des questions !

