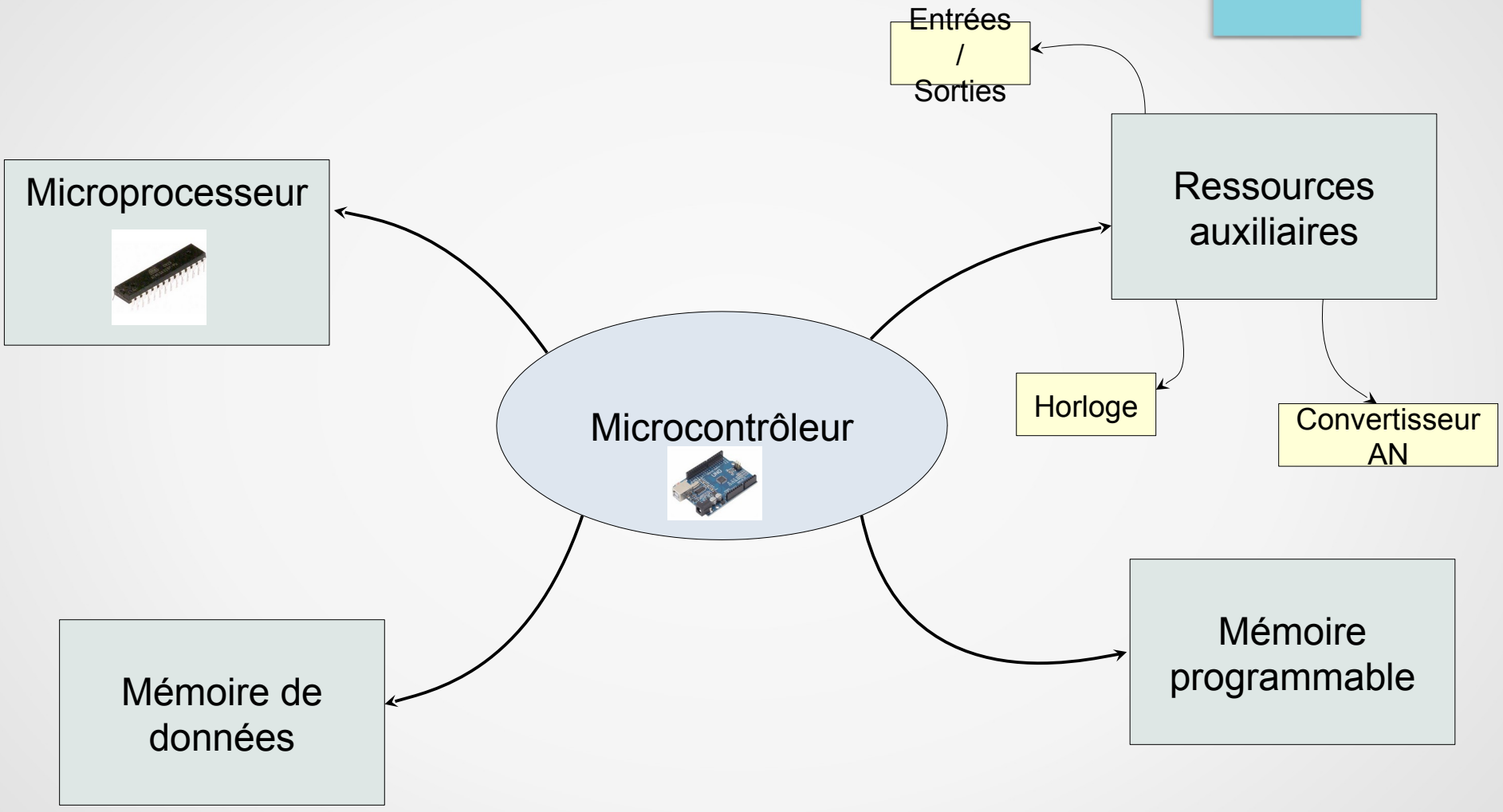
CEPEC 2021 – Formation Arduino™

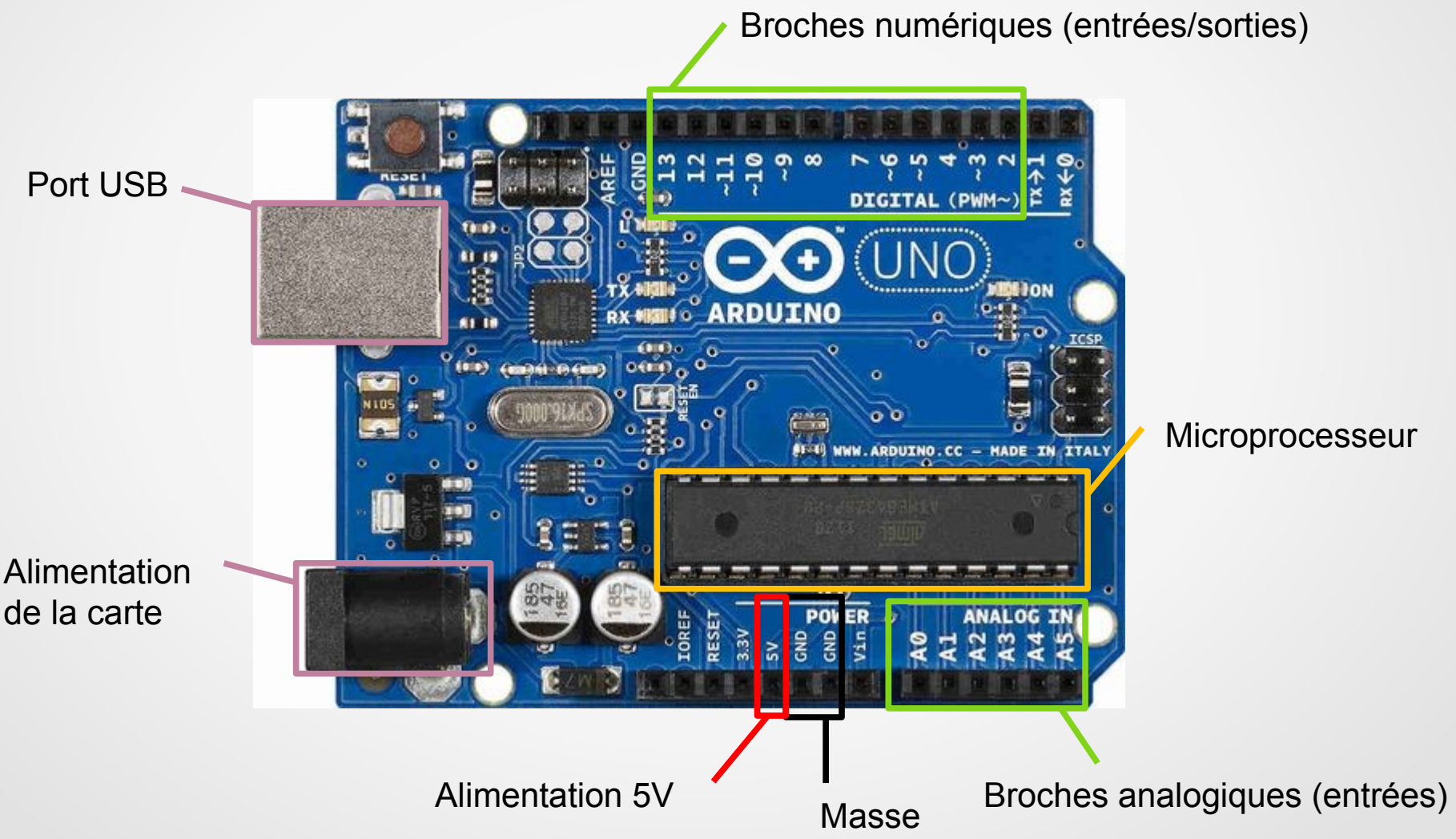
Qu'est-ce que c'est un microcontrôleur ?



I. L'environnement Arduino™
1. La carte Arduino™

microcontroleur 8 bits ATMEGA328P cadencé à 16 Mhz ;
alimentation externe (7 à 12 V) ou USB (5 V) ;
programmation et communication via port USB ;
14 broches d’entrées/sorties numériques dont 6 PWM ;
6 entrées analogiques sur 10 bits ;
1 port I2C (communication avec capteurs/actionneurs numériques) ;
1 port UART (communication série) ;
3 timers (comptage et mesure de temps) ;
gestion des interruptions.
une mémoire flash de 32 KB
un de SRAM 2 KB
un EPPROM de 1 KB

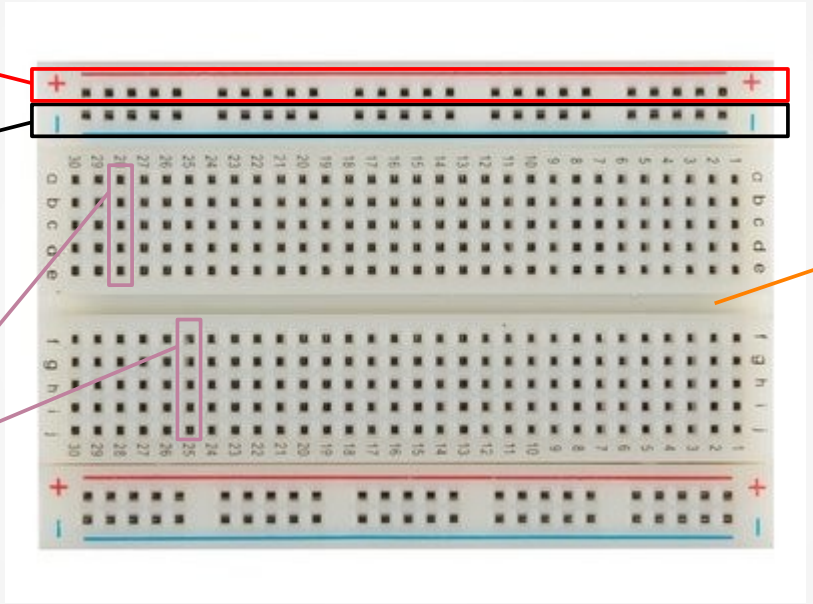
I. L'environnement Arduino™
1. La carte Arduino™



Ligne de trous reliés entre eux et servant généralement à l'alimentation

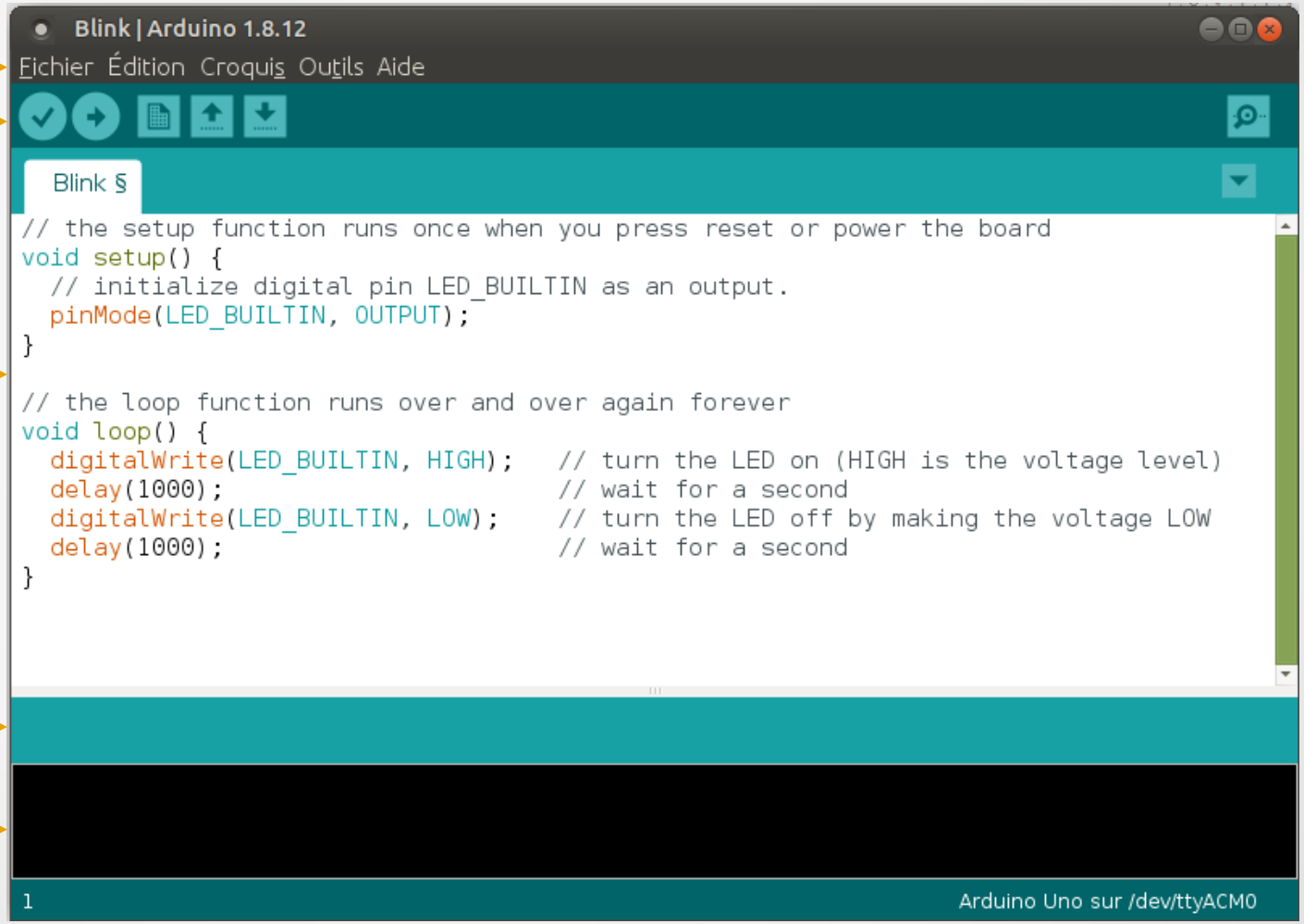
Ligne de trous reliés entre eux et servant généralement à la masse

Tous les point d'une même colonne sont reliés entre eux



Séparation de la plaque en deux parties symétriques et indépendantes

I. L'environnement Arduino™
3. Le logiciel



Barre de menu

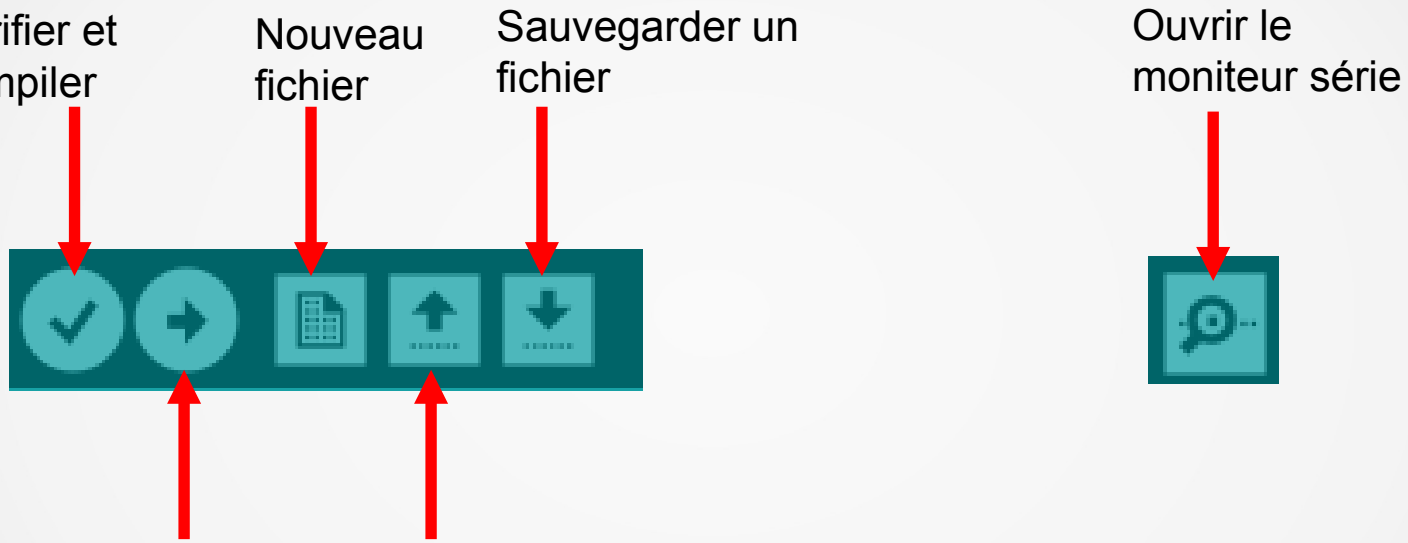
Barre des boutons

Fenêtre d'édition des programmes

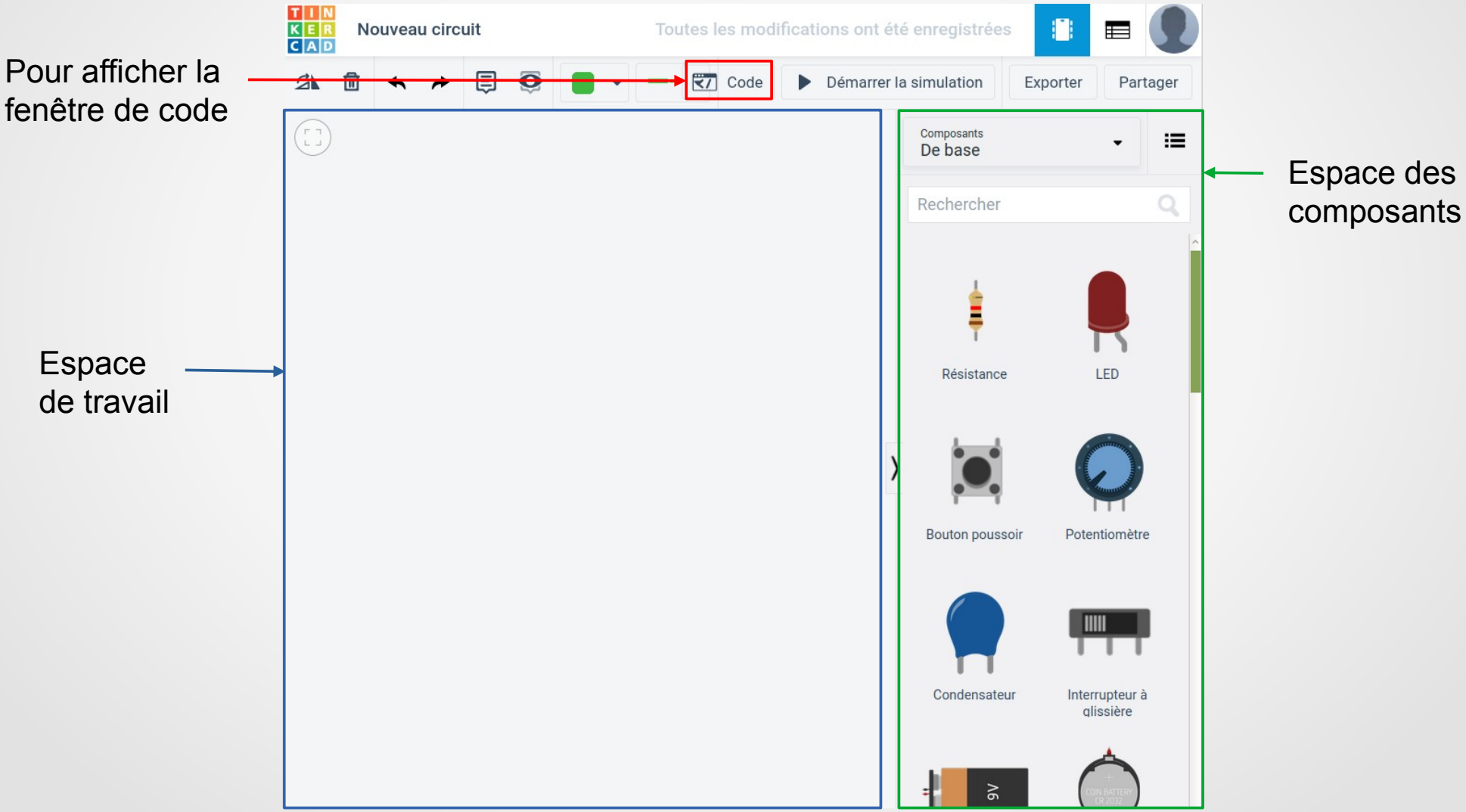
Zone de message

Console d'affichage des messages de compilation

I. L'environnement Arduino™
3. Le logiciel



I. L'environnement Arduino™
3. Le logiciel



I. L'environnement Arduino™
3. Le logiciel

Pivoter le composant

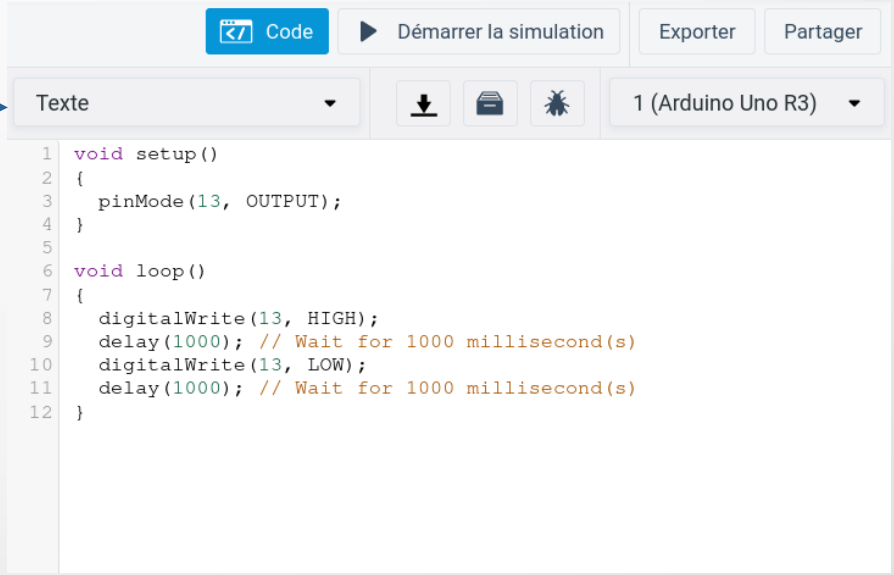
Afficher ou non les commentaires

Type de fils

Mettre un commentaire

Couleur des fils

Choisir *texte*, *bloc* fait de la programmation en scratch



I. L'environnement Arduino™
4. La structure d'un programme

Zone de définition des variables, des constantes, d'attribution des broches et d'appel des bibliothèques

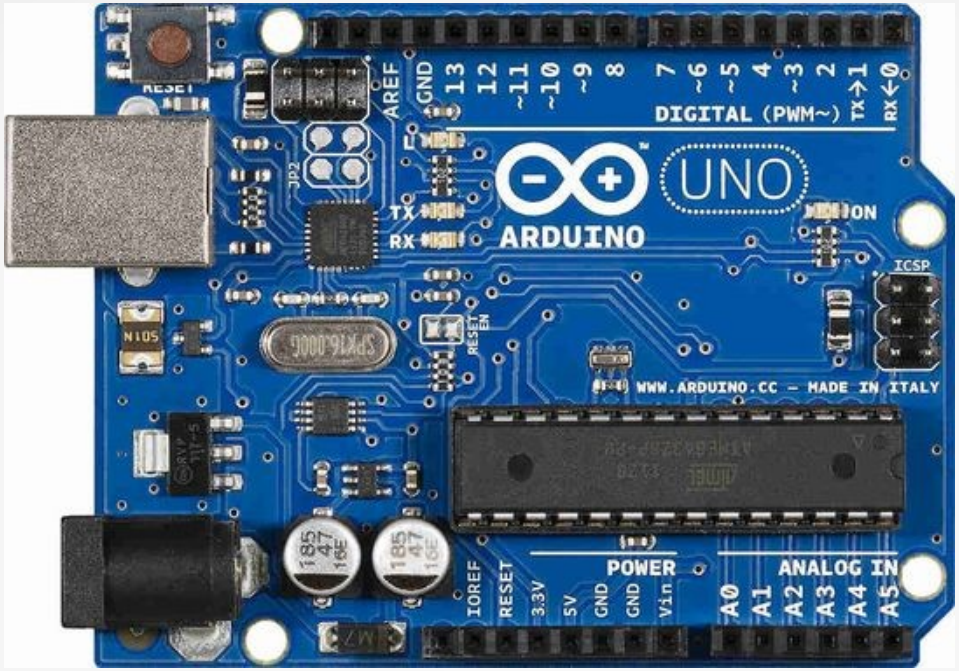
```
/* Programme qui fait clignoter une LED branchée sur la broche 4 */
int LedPin = 4;    // La LED est branchée sur la broche 4

// the setup function runs once when you press reset or power the board
void setup()
{
  pinMode(LedPin, OUTPUT);     // Définit la broche 4 comme une sortie.
}

// the loop function runs over and over again forever
void loop()
{
  digitalWrite(LedPin, HIGH);         // Met le sortie LedPin au niveau haut
  delay(1000);                        // attend 1 seconde
  digitalWrite(LedPin, LOW);          // Met le sortie LedPin au niveau bas
  delay(1000);                        // attend 1 seconde
}
```

Fonction *setup()* qui contient toutes les instructions d'initialisation
Cette partie ne s'exécute qu'une fois en début de programme

Fonction loo*p()* qui contient toutes les instructions du programme
Cette partie s'exécute en boucle

II. Manipuler l'Arduino™

1. Configuration en sorties numériques



La configuration en mode entrée (INPUT) ou en mode sortie (OUTPUT) se fait avec la fonction `pinMode()`.

Rôle : Configurer une broche du microcontrôleur
Syntaxe : `pinMode(pin,mode)`
paramètres :
`pin` : numéro ou nom de la broche à configurer
`mode` : INPUT, OUTPUT ou INPUT_PULLUP
renvoie : rien

II. Manipuler l'Arduino™

1. Configuration en sorties numériques

Application : Le programme Blink

Ce programme permet de faire clignoter une led avec une fréquence de 0,5Hz.

```
// the setup function runs once when you press reset or power the board
void setup() {
        // initialize digital pin LED_BUILTIN as an output.
        pinMode(LED_BUILTIN, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {
        digitalWrite(LED_BUILTIN, HIGH);    // turn the LED on (HIGH is the voltage level)
        delay(1000);                        // wait for a second
        digitalWrite(LED_BUILTIN, LOW);     // turn the LED off by making the voltage LOW
        delay(1000);                        // wait for a second
}
```

Le niveau électrique d'une broche configurée en sortie est fixée par le microcontrôleur avec la fonction : `digitalWrite()`.

Rôle : Affecter un état à une broche (en sortie) du microcontrôleur

Syntaxe : `digitalWrite(pin,value)`

paramètres :

`pin` : numéro ou nom de la broche à configurer

`value` : HIGH ou LOW

renvoie : `rien`

II. Manipuler l'Arduino™

1. Configuration en sorties numériques

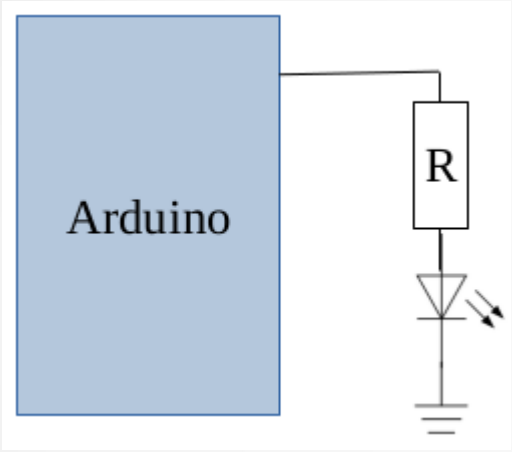


TABLE INTERNATIONALE DES VALEURS NORMALISEES DES RESISTANCES A ± 10 %						
ohm		Kiloohm			mégohm	
10	100	1	10	100	1	10
12	120	1,2	12	120	1,2	12
15	150	1,5	15	150	1,5	15
18	180	1,8	18	180	1,8	18
22	220	2,2	22	220	2,2	22
27	270	2,7	27	270	2,7	
33	330	3,3	33	330	3,3	
39	390	3,9	39	390	3,9	
47	470	4,7	47	470	4,7	
56	560	5,6	56	560	5,6	
68	680	6,8	68	680	6,8	
82	820	8,2	82	820	8,2	

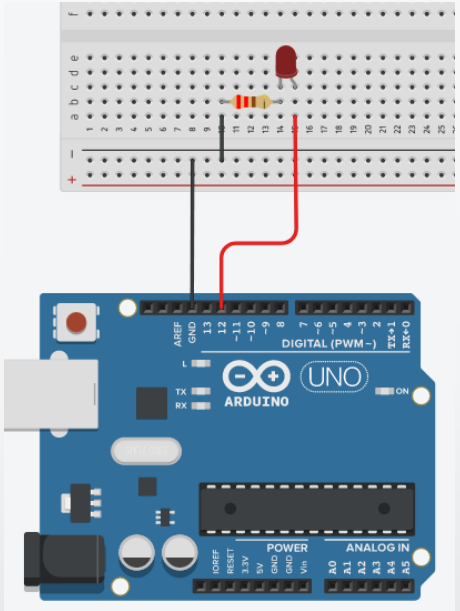
Fig. 13. - Valeurs des résistances avec tolérances ±10 %

Choisir, dans le tableau précédent, une résistance de protection de sorte que la valeur de l'intensité du courant I reste inférieure à 20 mA.

II. Manipuler l'Arduino™

1. Configuration en sorties numériques

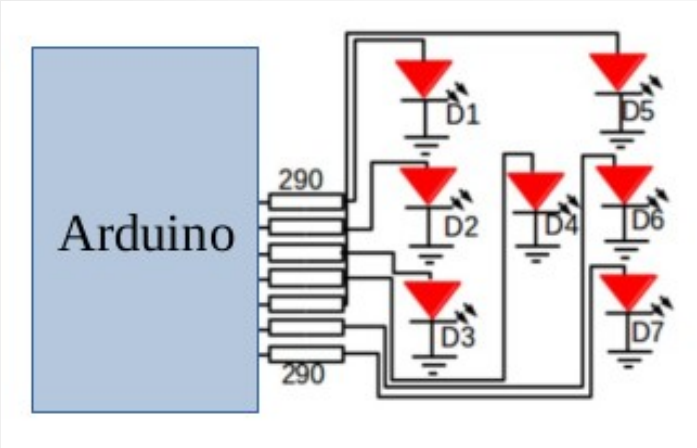
```
// the setup function runs once when you press reset or power the board
void setup() {
        // initialize digital pin LED_BUILTIN as an output.
        pinMode(LED_BUILTIN, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {
        digitalWrite(LED_BUILTIN, HIGH);    // turn the LED on (HIGH is the voltage level)
        delay(1000);                                    // wait for a second
        digitalWrite(LED_BUILTIN, LOW);     // turn the LED off by making the voltage LOW
        delay(1000);                                    // wait for a second
}
```



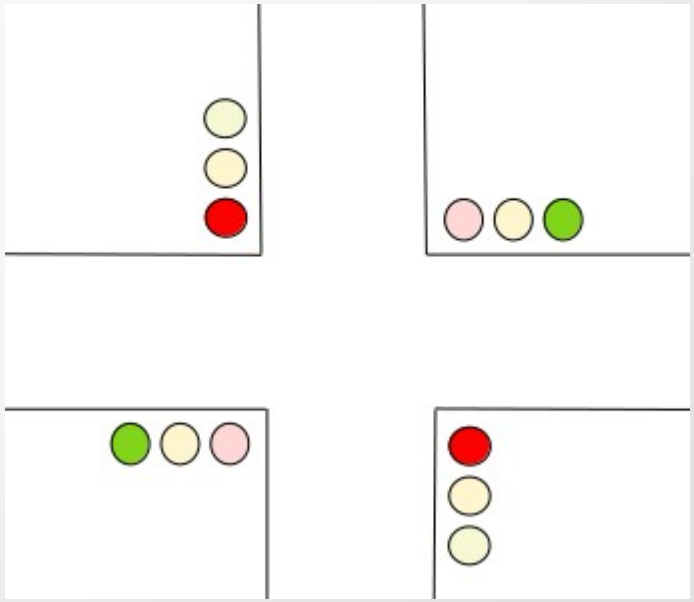
II. Manipuler l'Arduino™

1. Configuration en sorties numériques

Exemple en autonomie : Le Dé lumineux



Autre exemple en autonomie : Le feu de croisement



Le niveau électrique d'une broche configurée en entrée est lue par le microcontrôleur par la fonction `digitalRead()`.

Rôle : Lire l'état d'une broche (en entrée) du microcontrôleur
Syntaxe : `digitalRead(pin)`
paramètres :
`pin` : numéro ou nom de la broche à lire
renvoie : HIGH ou LOW

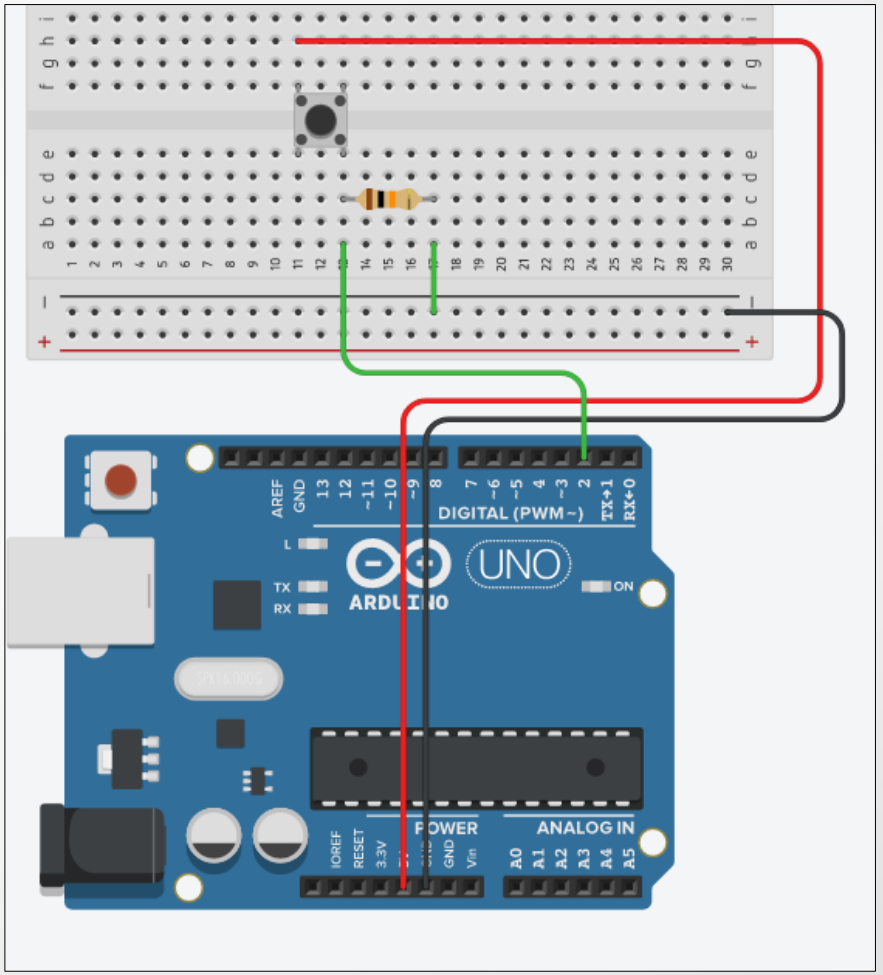
II. Manipuler l'Arduino™

2. Configuration en entrées numériques

Application : Le programme Button

Ce programme permet d'allumer la led en fonction de l'état d'un bouton.

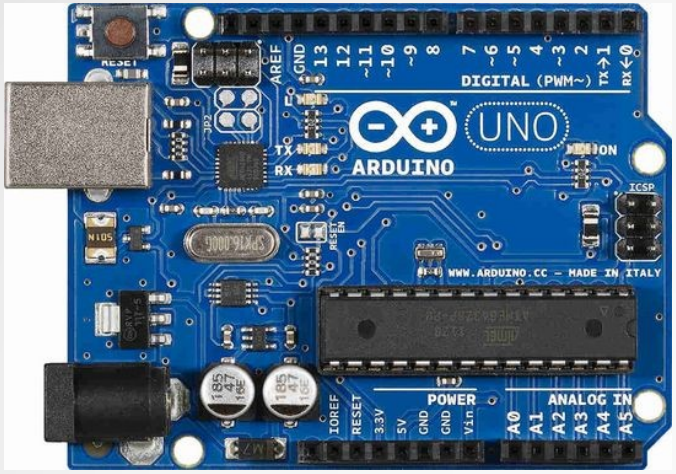
```
// constants won't change.
const int buttonPin = 2;
const int ledPin =  13;

// variables will change:
int buttonState = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}

void loop() {
  buttonState = digitalRead(buttonPin);

  if (buttonState == HIGH) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
}
```



Exemple en autonomie : commande avec 2 boutons poussoirs
Lire l'état de 2 boutons-poussoirs connectés aux broches 8 et 9 pour allumer ou éteindre une led connectée à la broche 5.

II. Manipuler l'Arduino™

3. Configuration en entrées analogiques



Une entrée analogique est un voltmètre : la carte mesure la différence de potentiel entre le fil qui est connecté sur l'entrée analogique et le potentiel du port GND

Le microcontrôleur ne travaille qu'avec des chiffres : il va donc convertir la tension mesurée en un nombre.

Le CAN de la carte Arduino™ travaille sur 10 bits : il accepte en entrée une tension comprise entre 0V et V_{ref}, et fournit au microcontrôleur un chiffre entier compris entre 0 et 1023 .

– Pour une tension de 0V (ou moins), le CAN retourne la valeur 0.

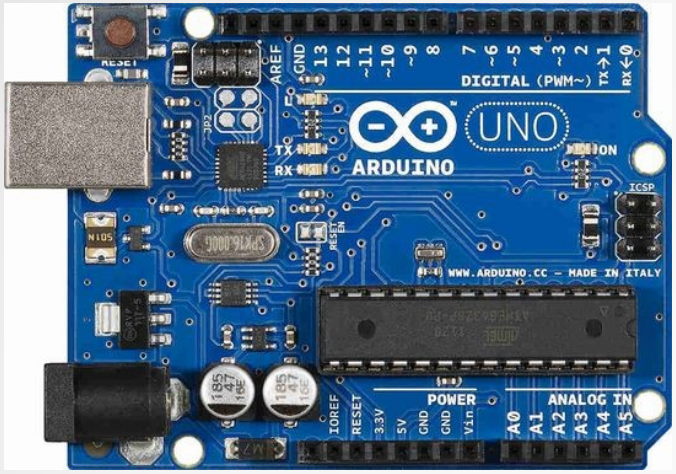
– Pour une tension V_{ref} (ou plus), le CAN retourne la valeur 1023.

– Pour une tension intermédiaire, le CAN retourne un entier compris entre 0 et 1023, en suivant une loi de proportionnalité.

La tension V_{ref} est 5 V par défaut

II. Manipuler l'Arduino™

3. Configuration en entrées analogiques



La lecture de la valeur numérique résultat de la conversion A/N de la tension analogique présente sur le canal spécifié est réalisée par la fonction `analogRead()`.

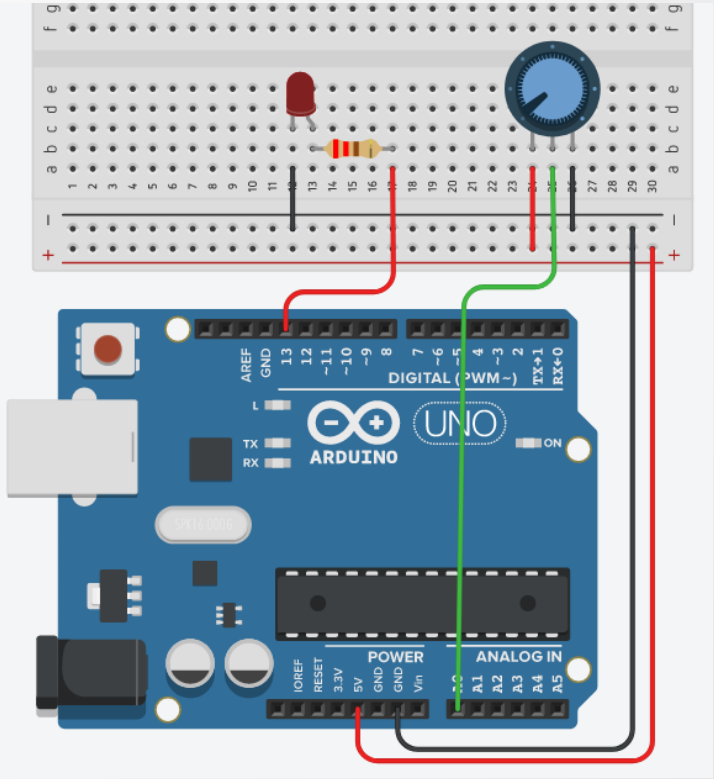
Rôle : Lire la valeur numérique (après conversion AN) d'une entrée analogique du microcontrôleur
Syntaxe : `analogRead(pin)`
paramètres :
`pin` : numéro ou nom de la broche à lire
renvoie : une valeur numérique entière

II. Manipuler l'Arduino™

3. Configuration en entrées analogiques

Application : Le programme AnalogInput
Ce programme permet de faire clignoter la led en fonction de la position du potentiomètre.

```
int sensorPin = A0;
int ledPin = 13;
int sensorValue = 0;

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

void loop() {
        sensorValue = analogRead(sensorPin);
        digitalWrite(ledPin, HIGH);
        delay(sensorValue);
        digitalWrite(ledPin, LOW);
        delay(sensorValue);
}
```

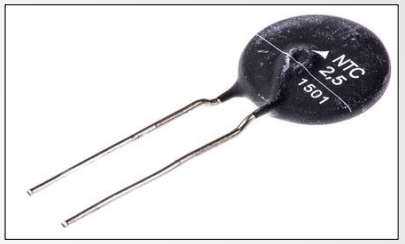


Exemple en autonomie : Thermistance

Une thermistance est une résistance thermique: sa valeur varie en fonction de la température.

Il existe deux types: NTC (negative temperature coefficient) et PTC (positive temperature coefficient). En general, on utilise des NTC.

Pour le câblage, la thermistance est associée en série avec une résistance de 10kOhm.



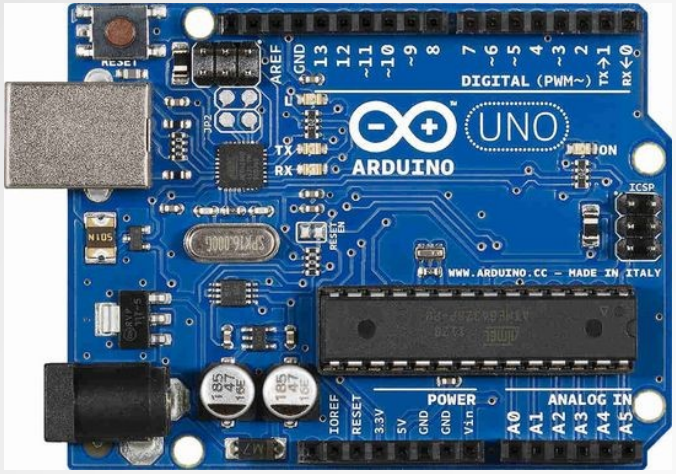
Autre exemple en autonomie : Photorésistance

Un photorésistance appelée également LDR « Light dependant resistor » est une résistance dont la valeur dépend de la quantité de lumière reçue par le composant. Cette résistance prend une valeur de 50kW en obscurité et varie jusqu'à 500W en pleine lumière. Pour convertir cette variation de résistance en variation de tension, il faut bien entendu apporter une alimentation.

Pour le câblage, la photorésistance est associée en série avec une résistance de 10kOhm.

II. Manipuler l'Arduino™

4. Configuration pour la communication numérique



Le microcontrôleur peut échanger des informations avec d'autres microcontrôleurs ou périphériques (écrans, capteurs numériques, ...) via des communications série.

La liaison série est configurée par la fonction `Serial.begin()` qui permet de fixer la vitesse de transmission (en bits par seconde bps) des informations.

Rôle : ouvre et définit la vitesse de la liaison série
Syntaxe : `Serial.begin(speed)`
paramètres :
`speed` : vitesse de transmission en bit/s (baud) (généralement 96000)
renvoie : rien

L'information est convertie en une chaîne de caractère puis émise sur la liaison série par la fonction : `Serial.print()`

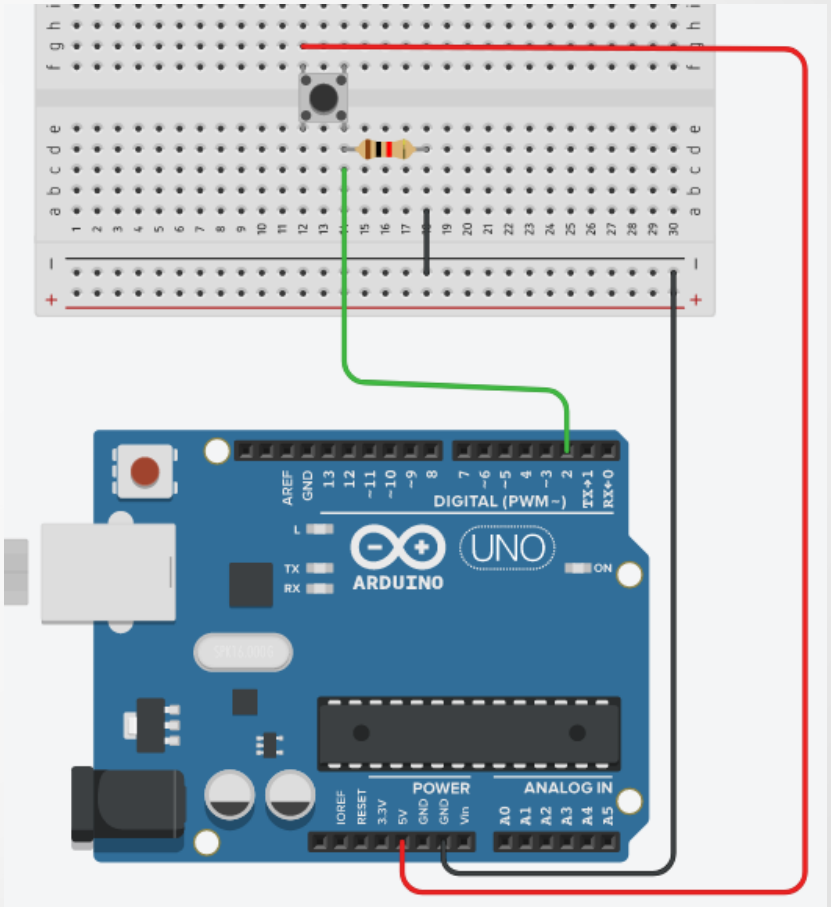
Rôle : Lire la valeur numérique (après conversion AN) d'une entrée analogique du microcontrôleur
Syntaxe : `Serial.print(valeur)`
paramètres :
`valeur` : la valeur à imprimer, tout type de valeurs (pour du texte, mettre entre " ")
renvoie : la valeur sur le moniteur série

II. Manipuler l'Arduino™

4. Configuration pour la communication numérique

Application Suivi d'appui sur un bouton
Ce programme renvoie 1 ou 0 selon qu'on appuie sur le bouton et l'affiche sur le moniteur série.

```
int pushButton = 2;

void setup() {
   Serial.begin(9600);
   pinMode(pushButton, INPUT);
}

void loop() {
  int buttonState = digitalRead(pushButton);
   Serial.println(buttonState);
 delay(1);
}
```



III. Utilisation en sciences physiques
1. Que disent les porgrammes ?

En seconde

Signal sonore périodique, fréquence et période.	Définir et déterminer la période et la fréquence d'un signal sonore notamment à partir de sa représentation temporelle. ***Utiliser une chaîne de mesure pour obtenir des informations sur les vibrations d'un objet émettant un signal sonore.*** ***Mesurer la période et la fréquence d'un signal sonore périodique.*** ***Utiliser un dispositif comportant un microcontrôleur pour produire un signal sonore.***
Capteurs électriques.	Citer des exemples de capteurs présents dans les objets de la vie quotidienne. ***Mesurer une grandeur physique à l'aide d'un capteur électrique résistif. Produire et utiliser une courbe d'étalonnage reliant la résistance d'un système avec une grandeur d'intérêt (température, pression, intensité lumineuse, etc.).*** ***Utiliser un dispositif avec microcontrôleur et capteur.***

III. Utilisation en sciences physiques
1. Que disent les porgrammes ?

En première spécialité

Modèle de comportement d'un gaz : loi de Mariotte.	Utiliser la loi de Mariotte. ***Tester la loi de Mariotte, par exemple en utilisant un dispositif comportant un microcontrôleur.***
Célérité d'une onde. Retard.	Exploiter la relation entre la durée de propagation, la distance parcourue par une perturbation et la célérité, notamment pour localiser une source d'onde. ***Déterminer, par exemple à l'aide d'un microcontrôleur ou d'un smartphone, une distance ou la célérité d'une onde.*** ***Illustrer l'influence du milieu sur la célérité d'une onde.***
Absorbance, spectre d'absorption, couleur d'une espèce en solution, loi de Beer-Lambert.	Expliquer ou prévoir la couleur d'une espèce en solution à partir de son spectre UV-visible. Déterminer la concentration d'un soluté à partir de données expérimentales relatives à l'absorbance de solutions de concentrations connues. ***Proposer et mettre en œuvre un protocole pour réaliser une gamme étalon et déterminer la concentration d'une espèce colorée en solution par des mesures d'absorbance. Tester les limites d'utilisation du protocole.***
Couleur blanche, couleurs complémentaires.	Choisir le modèle de la synthèse additive ou celui de la synthèse soustractive selon la situation à interpréter.

III. Utilisation en sciences physiques
1. Que disent les porgrammes ?

En première enseignement scientifique

Projet expérimental et numérique

Le projet s'articule autour de la mesure et des données qu'elle produit, qui sont au cœur des sciences expérimentales. L'objectif est de confronter les élèves à la pratique d'une démarche scientifique expérimentale, de l'utilisation de matériels (capteurs et logiciels) à l'analyse critique des résultats.

Le projet expérimental et numérique comporte trois dimensions :

- utilisation d'un capteur éventuellement réalisé en classe ;
- acquisition numérique de données ;
- traitement, représentation et interprétation de ces données.

Selon les projets, l'une ou l'autre de ces dimensions peut être plus ou moins développée.

L'objet d'étude peut être choisi librement, en lien avec le programme ou non. Il s'inscrit éventuellement dans le cadre d'un projet de classe ou d'établissement. Le travail se déroule sur une douzaine d'heures, contiguës ou réparties au long de l'année. Il s'organise dans des conditions matérielles qui permettent un travail pratique effectif.

La dimension numérique repose sur l'utilisation de matériels (capteur éventuellement associé à un microcontrôleur) et de logiciels (tableur, environnement de programmation).

III. Utilisation en sciences physiques
1. Que disent les porgrammes ?

En terminale spécialité

capacité d'un condensateur.	***Illustrer qualitativement, par exemple à l'aide d'un microcontrôleur, d'un multimètre ou d'une carte d'acquisition, l'effet de la géométrie d'un condensateur sur la valeur de sa capacité.***
Capteurs capacitifs.	Expliquer le principe de fonctionnement de quelques capteurs capacitifs. ***Étudier la réponse d'un dispositif modélisé par un dipôle RC.*** ***Déterminer le temps caractéristique d'un dipôle RC à l'aide d'un microcontrôleur, d'une carte d'acquisition ou d'un oscilloscope.***

III. Utilisation en sciences physiques
2. Quel intérêt pour les sciences physiques ?

Les élèves peuvent concevoir eux-même les outils d'une expérimentation.

- Notions de mesures, incertitudes, conversions analogique/numérique.

- Chaque étape d'une expérience peut être comprise et modifiée par les élèves (anti « boite noire »).

- Permet le lien entre les phénomènes physiques et un langage de programmation.

- Permet de réinvestir les connaissances en électricité de manière plus ludique.

- Il existe une multitude de capteurs et d'activités possibles à partir d'une même carte.

En revanche :
- les cartes microcontrôleurs ne remplacent pas les interfaces d'acquisitions performantes, déjà utilisées au lycée.

III. Utilisation en sciences physiques
3. Application en travaux pratiques

Déroulement d'une activité expérimentale avec microcontrôleur

→Réaliser le circuit
→Recopier, écrire ou compléter le code et le valider
→Téléverser le code
→Vérifier que le code commande correctement le circuit
→Commenter , modifier les paramètres, exploiter…

Difficultés à gérer / solutions

- Composants petits donc difficiles à manipuler → Le montage est donné
- Vérification des connexions sur la platine → Le montage est donné
- Vérification du code → Le code est donné
- Niveau hétérogène → Différentiation du sujet
- Gestion du groupe → Les élèves ayant terminé rapidement aident les autres

Organisation de l'activité expérimentale en fonction de la classe

- Préparation de l'activité expérimentale à la maison
- Faire réaliser le circuit / Donner le circuit
- Faire écrire le code / Donner le code (en entier ou en partie)
- Proposer différents niveaux

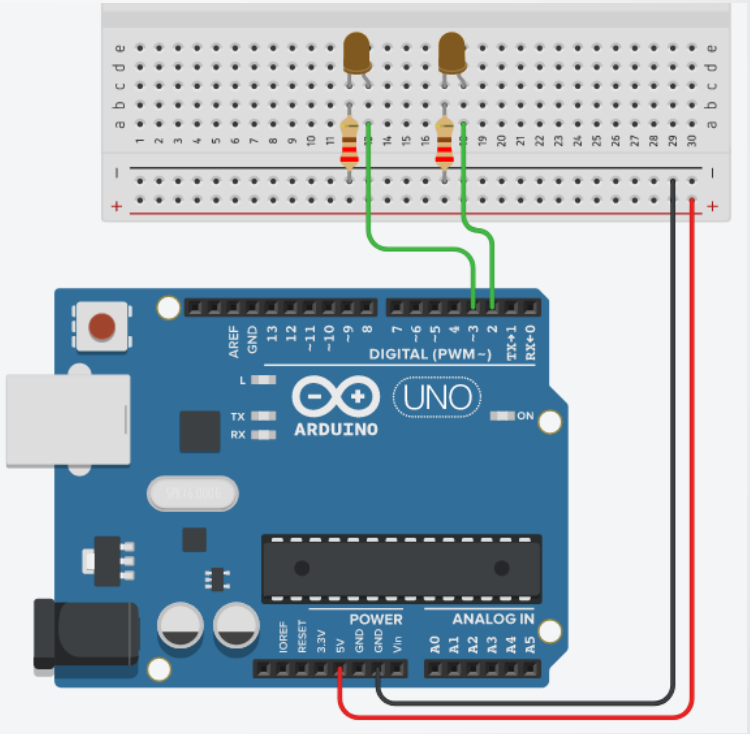
Quid de l'évaluation ?

Exemple d'évaluation

Afin de réaliser une décoration pour Noël à partir de deux petites maisons en bois, on ajoute une LED pour éclairer chaque maison de l'intérieur. On souhaite que les deux LED clignotent à une fréquence de 0,5 Hz et s'allument alternativement. On décide pour cela d'utiliser un microcontrôleur Arduino™.

Le programme pour contrôler les LED :

```
int ledOrange1= 2;       //indique que la LED 1 est branchée sur la broche 2
int ledOrange2 = 4;      //indique que la LED 2 est branchée sur la broche 4
/* On stocke la valeur 2 dans la variable "ledOrange1",
pour indiquer que la led orange notée 1 se trouve sur la broche 2,
même chose pour la led orange notée 2 sur la broche 3 */
void setup() {
pinMode(ledOrange1, OUTPUT);
// indique que la broche sur laquelle est branchée la LED 1 est une sortie
pinMode(ledOrange2, OUTPUT);
// indique que la broche sur laquelle est branchée la LED2 est une sortie
}
void loop() {
digitalWrite(ledOrange1,HIGH);
delay(1000);
digitalWrite(ledOrange1,LOW);
digitalWrite(ledOrange2,HIGH);
delay(100);
digitalWrite(ledOrange2,LOW);
}
```



1) D'après le montage réalisé, chaque LED a-t-elle été déclarée sur la broche à laquelle elle est connectée ? Si non, surligner la(ou les) ligne(s) de code fausse(s) et la (ou les) réécrire en la (ou les) corrigeant.
2) Quelle est la partie du code qui se répète tant que l'Arduino™ est branché ?
3) Ajouter un commentaire pour expliquer chaque ligne de code de la partie `void loop()`.
4) Il y a un problème, les LED ne clignotent pas avec la bonne fréquence. Surligner les lignes à corriger, puis les réécrire de manière à ce que les LED clignotent avec la fréquence attendue. Justifier par un calcul.

III. Utilisation en sciences physiques
4. Exemples de travaux pratiques

Seconde : Période et fréquence d'un signal sonore

Objectifs du TP

- Visualiser le signal correspondant au son produit par un diapason sur sa caisse de résonance
- Mesurer la période du signal et en déduire la fréquence du son.
- Produire un signal sonore avec un microcontrôleur

Contexte

Pour l'anniversaire de sa petite sœur, un élève voudrait la surprendre et faire en sorte que la chanson "joyeux anniversaire" se joue à l'ouverture de son cadeau. Il dispose d'un Arduino™ et compte sur vos compétences en physique et en programmation pour l'aider.

Seconde : Mesure de la température à l'aide d'une thermistance

Objectifs du TP

- Mesurer une température à l'aide d'une thermistance CTN
- Produire et utiliser une courbe d'étalonnage reliant la résistance de la thermistance à sa température.
- Utiliser un microcontrôleur Arduino™ associé à une thermistance pour afficher la température.

Contexte

Une élève souhaite faire chauffer de l'eau pour son thé. La variété qu'elle souhaite faire infuser doit l'être dans une eau à 70°C, pas plus. Elle souhaiterait réaliser un thermomètre électronique qui l'avertit quand l'eau et à la température souhaitée. Elle dispose d'un Arduino™ et compte sur vos compétences en physique et en programmation pour l'aider.

Première : Mesure de distance et célérité d'une onde

Objectifs du TP

Mesurer la célérité d'une onde sonore

Modifier un programme pour déterminer une distance à l'aide d'un microcontrôleur

Contexte

Un enfant possède un camion télécommandé et souhaiterait installer un radar de recul sur ce dernier. Son grand frère possède un Arduino™ et propose de lui fabriquer ce dispositif. Il sollicite votre aide pour collecter toutes les données nécessaires puis réaliser le dispositif complet.