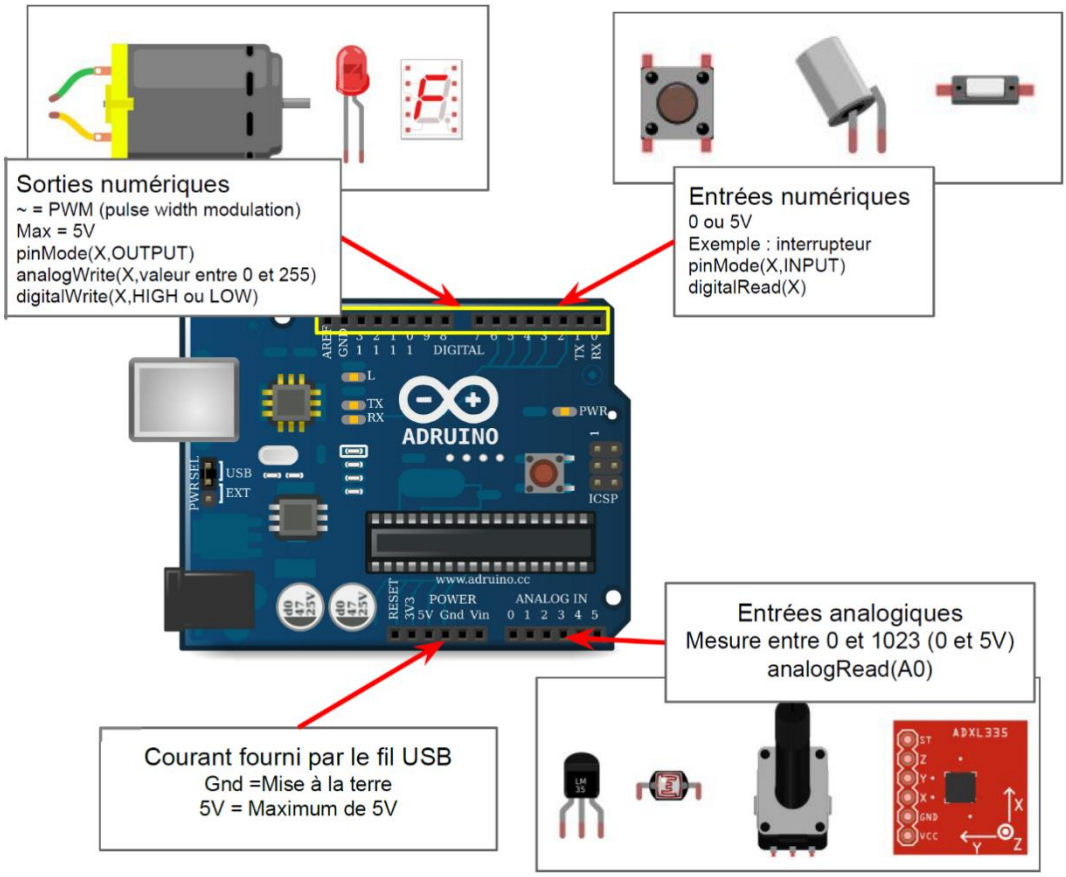
Se former à l'utilisation de microcontrôleurs pour la physique chimie

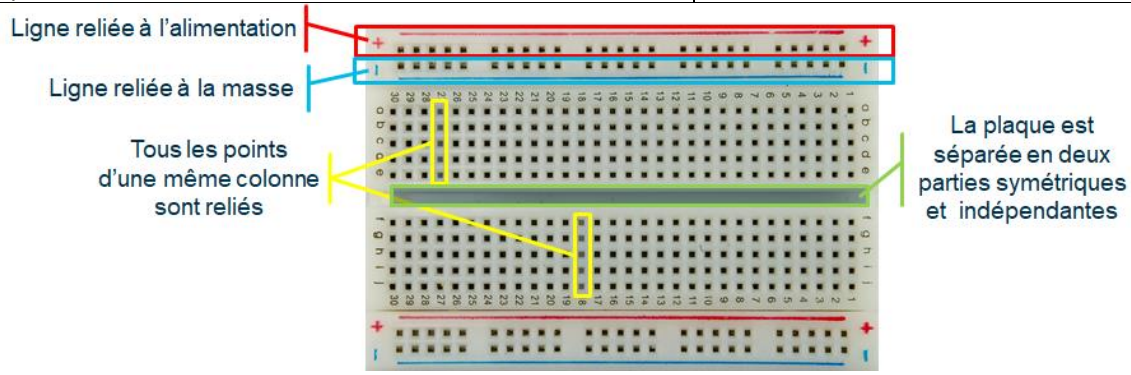
Le présent livret présente des outils concrets permettant la prise en main des microcontrôleurs dans le cadre des nouveaux programmes de physique-chimie, en classe de seconde et en spécialité de la classe de première. Le microcontrôleur choisi ici pour les exemples est Arduino™ mais peut être adapté à tout autre microcontrôleur.

Ce livret a été conçu initialement par Cécile Cathala, relu et amendé par Patricia Marchand et Jacques Vince.

1. Le matériel



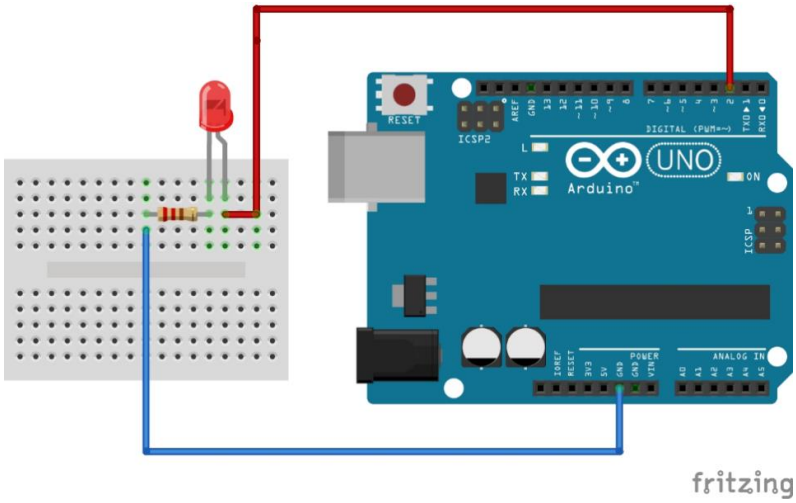
Entrées (INPUT) : Lues par le µC pour connaître l'état du système auquel il est relié	Sorties (OUTPUT) : Déclenchement contrôlé par le µC => Source de tension
Entrée numérique (DIGITAL) (ports 2 à 13) : lit l'état du port (LOW ou HIGH) *Exemple : interrupteur*	Sortie numérique (DIGITAL) (ports 2 à 13) : génère une tension de 0V (LOW) à 5V (HIGH) *Exemples : moteur, LED, buzzer*
Entrée analogique (ports A0 à A5) => Voltmètre qui mesure une tension entre 0 et 5V (10 bits donc 1023 valeurs possibles puis CAN) *Exemples : photorésistance, thermistance*	



2. Quelques programmes d'appropriation

2.1. Allumer une LED

Niveau 1 : Brancher la LED sur la bonne broche, téléverser le programme.




```
const int LedRouge = 2;

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

void loop() {
digitalWrite(LedRouge, HIGH);
}
```

Niveau 2 : Modifier le programme de sorte à changer la sortie de la LED, par exemple sur la broche 5. Enregistrer et tester.

2.2. Stroboscope

Niveau 1 : Brancher la LED sur la bonne broche, téléverser le programme.

Montage identique au précédent

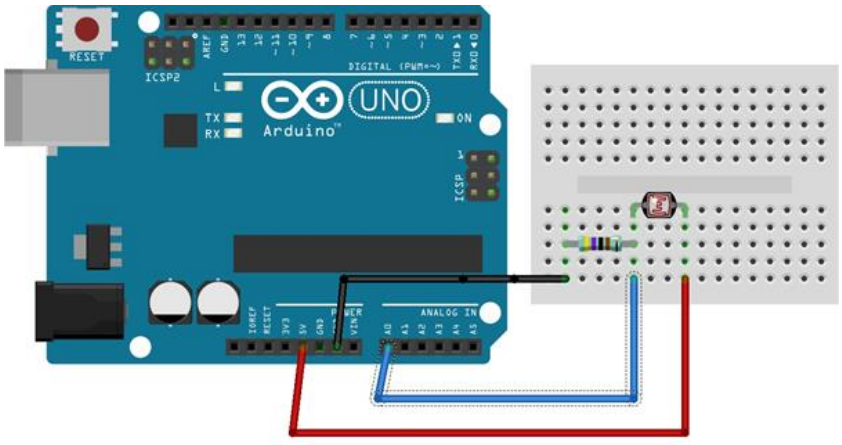
```
const int LedRouge = 2;

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

void loop() {
	digitalWrite(LedRouge, HIGH);
	delay(500) ;
	digitalWrite(LedRouge, LOW);
	delay(500);
 }
```

Niveau 2 : Modifier le programme de sorte à modifier la durée de clignotement par exemple 50 ms.

Niveau 3 : Modifier le programme de sorte à faire clignoter alternativement une LED rouge et une LED verte.

2.3. Capteur de lumière

Niveau 1 : Brancher la photorésistance sur l'entrée analogique A0, téléverser le programme puis ouvrir le moniteur série (bouton en haut à droite de la fenêtre).



```
const int photoresistance = A0;
float valeur = 0 ;

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

void loop() {
 valeur=analogRead(photoresistance);
 Serial.print("Valeur mesurée : ");
 Serial.println(valeur) ;
 delay(1000);
}
```

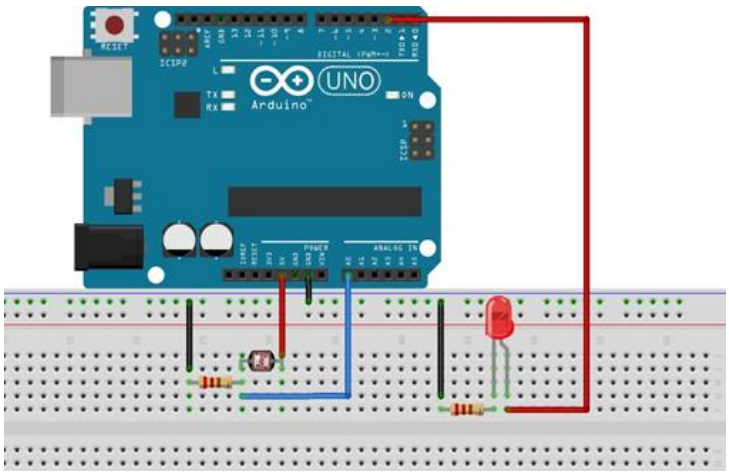
Niveau 2 : Modifier le programme de sorte à diminuer la durée entre deux mesures à 100 ms au lieu de 1000 ms. Téléverser puis dans l'onglet "Outils", cliquer sur "Traceur série" afin d'afficher la courbe de mesure.

Niveau 3 : La grandeur mesurée par la photorésistance est stockée sur 10 bits, ce qui explique qu'elle prenne des valeurs entre 0 et 1023 (2^{10} possibilités).

En réalité, ces valeurs correspondent à 1024 valeurs de tension aux bornes de la résistance comprises entre 0 et 5 V. Compléter les lignes du programme de sorte à afficher dans le moniteur série la tension aux bornes de la photorésistance.

2.4. Allumage automatique

Niveau 1 : Brancher la photorésistance et la LED sur les entrées adéquates, téléverser le programme et tester la sensibilité de l'allumage de la LED.

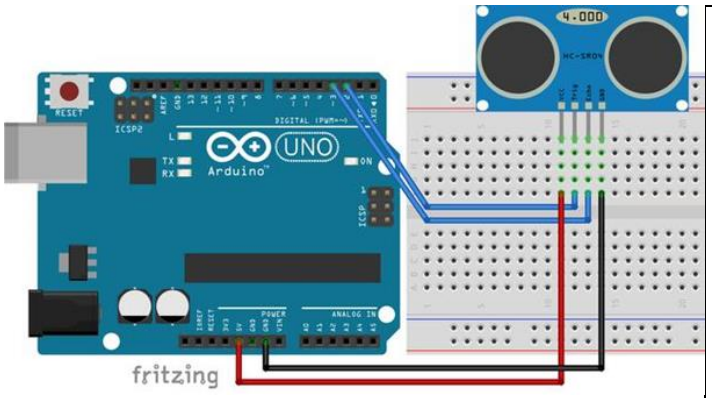


```
const int photoresistance = A0;
const int LEDrouge = 2;
int valeur = 0 ;

void setup() {
 pinMode(photoresistance, INPUT);
 pinMode(LEDrouge, OUTPUT);
 Serial.begin(9600);
}

void loop() {
 valeur = analogRead(photoresistance);
 if(valeur > 300){
       digitalWrite(LEDrouge, LOW);
 }
 else if(valeur <= 300){
       digitalWrite(LEDrouge, HIGH);
 }
}
```

Niveau 2 : Modifier la valeur seuil. Enregistrer et tester.

Niveau 3 : Modifier le programme de sorte à ajouter une condition supplémentaire : si la luminosité est encore plus faible, deux LED s'allument. Enregistrer, téléverser et tester.

2.5. Radar de recul



Broche VCC => 5 V
Broche Trig => pin2
Broche Echo => pin3
Broche GND => GND

Niveau 1 :

Réaliser les branchements de l'émetteur-récepteur d'ultrasons, téléverser le programme et ouvrir le moniteur série.

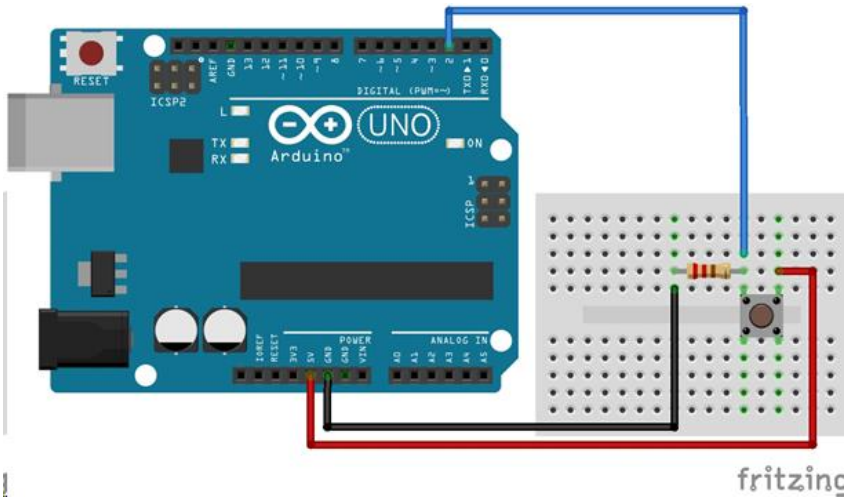
```
const int trig = 2;
const int echo = 3;
const unsigned long timeout = 25000UL;
const float sound_speed = 0.340;
long mesure = 0 ;
float distance = 0 ;

void setup() {
 Serial.begin(9600);
 pinMode(trig, OUTPUT);
 digitalWrite(trig, LOW);
 pinMode(echo, INPUT);
}
void loop() {
 digitalWrite(trig, HIGH);
 delayMicroseconds(10);
 digitalWrite(trig, LOW);
 mesure = pulseIn(echo, HIGH, timeout);
 distance = mesure / 2 * sound_speed;
 Serial.print("Attention,    l'obstacle    se
trouve à ");
 Serial.print(distance);
 Serial.println(" mm !");
 delay(300);
}
```

Niveau 2 : Brancher un buzzer sur la carte, lui assigner une broche dans le programme pour le faire bipper. Enregistrer et téléverser.

Niveau 3 : Ajouter quelques lignes de code à la fin de la boucle loop() de sorte à ce que le buzzer bippe lorsque l'obstacle détecté est trop près. Enregistrer et tester.

2.6. *Feu piéton*

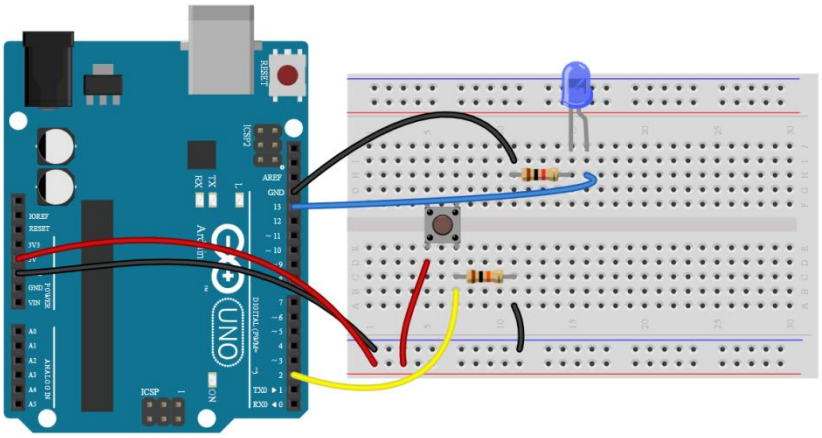
Niveau 1 : Le moniteur série renvoie l'information "Interrupteur fermé" si le bouton poussoir est enfoncé et renvoie l'information "Interrupteur ouvert" sinon.




```
const int bouton = 2;
int buttonState = 0;

void setup() {
  pinMode(bouton, INPUT);
  Serial.begin(9600);
}
void loop(){
buttonState = digitalRead(bouton);
  if (buttonState == HIGH) {
    Serial.println("interrupteur
fermé");
  }
   else {
    Serial.println("interrupteur
ouvert");}
    delay(100);
}
```

Niveau 2 :
Modifier le programme de sorte qu'une LED s'allume lorsque l'interrupteur est enfoncé.

Niveau 3 :
Modifier le programme de sorte qu'une LED verte s'allume si un bouton poussoir est enfoncé, reste allumée 5 secondes puis s'éteint une LED rouge s'allume jusqu'à ce qu'il y ait un nouvel appui sur le bouton.



3. Quelques exercices pour s'entrainer…

Exercice 1 **Feu tricolore**	**Exercice 2** **Feu tricolore et feu piéton**	**Exercice 3** **Ventilateur**
3 LED de couleurs différentes s'allument successivement (Rouge : 4 s - Vert - 6 s - Orange - 2 s)	Ajouter au défi précédent un feu piéton qui passe au vert pendant 5 secondes si le bouton poussoir est enfoncé et qui provoque le passage au rouge du feu tricolore.	Un ventilateur se met en route lorsque la température atteint une valeur limite définie par l'utilisateur

4. Mise en œuvre pratique

4.1. Proposition de matériel

Matériel de base pour les activités expérimentales avec microcontrôleurs (2nde et 1ère spécialité)

1 microcontrôleur	Arduino Uno, Adafruit, Funduino …	20 €
1 plaque de montage	Taille moyenne	4,50 €
1 câble ordi ⇔ carte long	(si ordi fixe)	3 €
Nappes de fils (jumpers) x 40	mâle / mâle	6 €
5 LED (rouges, vertes, oranges)	Rouge, verte, orange	1€
5 Résistances	220 Ω ou 1kΩ	0,20 €
1 thermistance	CTN 4,7 kΩ	0,60 €
1 buzzer	Deux broches	1,80 €
1 photorésistance	1 kΩ	0,75 €
1 émetteur récepteur US	Ex : HC SR 04	3,90 €
1 bouton poussoir		1,90 €
1 boîte plastique	Intercalaires amovibles	
1 capteur de pression	Ex : MPRLS	13 €
	TOTAL :	**Environ 60 €**

4.2. Sécurité

- Le microcontrôleur placé sur la carte est prévu pour fonctionner entre 3,3 et 5V.
- Le courant de sortie de chaque broche ne doit pas dépasser 40 mA.
- Le courant issu du port USB ne doit pas dépasser 500 mA.
- Dans le cas où les broches numériques du microcontrôleur sont des entrées (au lieu d'être raccordée à la masse, la broche sera raccordée au +5V), il faudra être très vigilant à ce que la broche soit configurée comme INPUT. Si elle devait être configurée en OUTPUT et réglée à 0V (LOW) par erreur, il est presque certain que le microcontrôleur sera endommagé.

Conseils de sécurité :

- Pour éviter qu'un fil ou qu'un composant branché au + vienne endommager un port USB dans l'ordinateur, isoler le métal du port USB avec un adhésif d'électricien
- Pour éviter les courts-circuits :
    - La carte ne doit pas être posée sur un support conducteur car elle possède sur son verso des zones nues qui ne doivent pas êtres mises en contact afin de ne pas court-circuiter les composants entre eux.
    - Ne jamais connecter directement le port noté « Gnd » (pôle négatif) avec la broche 5 V (pôle positif).
- **Pour protéger les entrées numériques :** connecter sur la patte du microcontrôleur utilisée comme INPUT une résistance d'une centaine d'Ohms qui limitera le courant en cas de fausse manœuvre.

Récapitulatif des commandes de base

Structure

FONCTIONS DE BASE

Ces deux fonctions sont obligatoires dans tout programme en langage Arduino :

- void setup()
- void loop()

STRUCTURES DE CONTRÔLE

- if (condition) {commande} ;
- if (condition) {commande} ; else {commande}
- for
- while

SYNTAXE DE BASE

- ; (point virgule)
- {} (accolades)
- // (commentaire sur une ligne)
- /* */ (commentaire sur plusieurs lignes)

OPÉRATEURS ARITHMÉTIQUES

- = (égalité)
- + (addition)
- - (soustraction)
- * (multiplication)
- / (division)
- % (modulo)

OPÉRATEURS DE COMPARAISON

- == (égal à)
- != (différent de)
- < (inférieur à)
- > (supérieur à)
- <= (inférieur ou égal à)
- >= (supérieur ou égal à)

Variables et constantes

Les variables sont des expressions que vous pouvez utiliser dans les programmes pour stocker des valeurs, telles que la tension de sortie d'un capteur présente sur une broche analogique.

CONSTANTES PRÉDÉFINIES

Les constantes prédéfinies du langage Arduino sont des valeurs particulières ayant une signification spécifique.

- HIGH | LOW
- INPUT | OUTPUT
- true | false

TYPES DES DONNÉES

Les variables peuvent être de type variés qui sont décrits ci-dessous.

- boolean
- int
- unsigned int
- long
- unsigned long
- float (nombres à virgules)
- Les chaînes de caractères
- void (fonctions)

Fonctions

ENTRÉES/SORTIES NUMÉRIQUES

- pinMode(broche, mode)
- digitalWrite(broche, valeur)
- int digitalRead(broche)

ENTRÉES ANALOGIQUES

- int analogRead(broche)

SORTIES "ANALOGIQUES" (GÉNÉRATION D'IMPULSION)

- analogWrite(broche, valeur) - PWM

ENTRÉES/SORTIES AVANCÉES

- tone(broche, fréquence, durée) ou tone (broche, fréquence)
- notone (broche, fréquence, durée) ou notone (broche, fréquence)
- pulseIn()

TEMPS

- delay(ms)
- delayMicroseconds(us)

MATH

- min(x, y)
- max(x, y)
- abs(x)
- sq(x)
- sqrt(x)

TRIGONOMÉTRIE

- sin(rad)
- cos(rad)
- tan(rad)
- degrees(rad)
- radians(deg)
- PI

COMMUNICATION

- Serial

Source : http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.Reference