PAR ETC – ARDUINO

ETC SCHOOL

EXERCICES D'APPLICATION

PROGRAMMATION ARDUINO

Auteur:

ETC School

12 Avril 2022

Travaux Pratiques Arduino
Programmation Industrielle

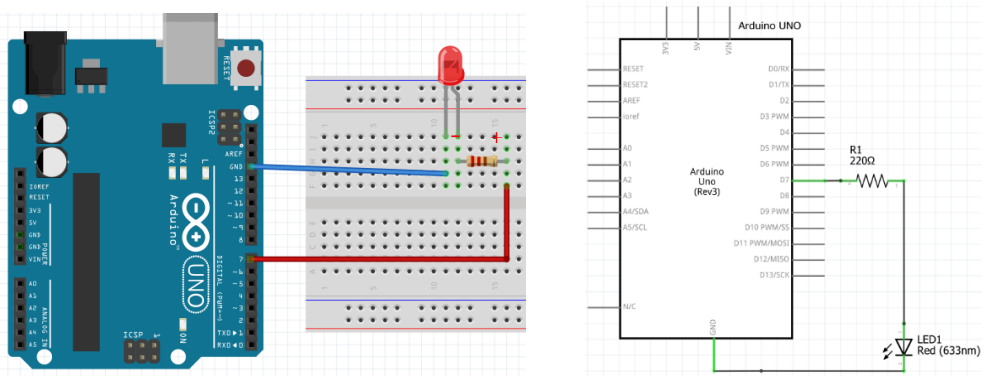
Exercice 1 : Faire clignoter une LED

Définition LED :

Une LED est un composant électronique qui permet d'émettre de la lumière lorsqu'elle est traversée par un courant électrique. Une LED est souvent utilisée dans le domaine de l'électronique pour indiquer visuellement une information.

Notes : Pour l'utilisation d'une LED, il est impératif d'utiliser une résistance en série pour limiter le courant afin d'éviter d'endommager la LED. En effet, cette résistance sert de protection à la LED.

1. Réaliser le schéma de câblage (ci-dessous) en utilisant de logiciel Fritzing.
   - La LED doit être connectée sur la broche 7.

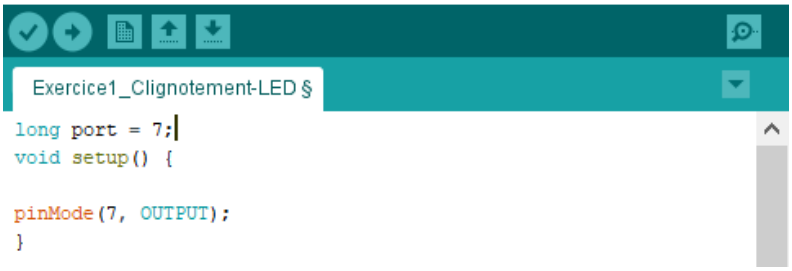


2. Câbler le montage sur la breadbord (attention à bien respecter le sens du montage de la LED) et 150Ω < R< 220 Ω.
3. Programmer la carte Arduino en utilisant le code ci-dessous :

```
Exercice1_Clignotement-LED §

/************************************/
/*   Déclarations des ports         */
/************************************/
long port = 7; //On déclare l'utilisation de la broche 7
/************************************/
/*   Initialisation du code         */
/************************************/
void setup() {

pinMode(7, OUTPUT); //On initialise la broche 7 comme étant une sortie numérique
}
/************************************/
/*        Boucle infinie            */
/************************************/
void loop() {
  digitalWrite(7,HIGH); //On allume la LED en envoyant un état haut (5V) sur la broche 7
  delay(1000);          //On garde la LED allumer pendant 1000 miliseconde = 1 seconde
  digitalWrite(7,LOW);  //On éteint la LED en envoyant un état bas (0V) sur la broche 7
  delay(1000);          // //On garde la LED éteinte pendant 1000 miliseconde = 1 seconde
}
```

Remarque : Si votre programme ne fonctionne pas, il est tout à fait normal car il y a une erreur sur le schéma de câble question 1 par rapport au programme. Corriger cette erreur puis relancer.

4. Commenter soigneusement votre code, puis enregistrer le sur votre bureau.
5. Maintenant que vous êtes un professionnel des clignotements des LED, connecter 4 autres LEDs, puis programmer l'Arduino de façon que ces LED s'allument les unes après les autres. Vous pouvez réduire la durée des états (HIGH/LOW) pour accélérer le clignotement.
6. Pour améliorer la lisibilité de votre code, prenez dès maintenant l'habitude de déclarer vos broches en faisant une affectation. Dans notre cas voir (l'image ci-dessous). Puis remplacer le sur votre code et exécuter le code. Cette affectation doit être faite avant la fonction principale.



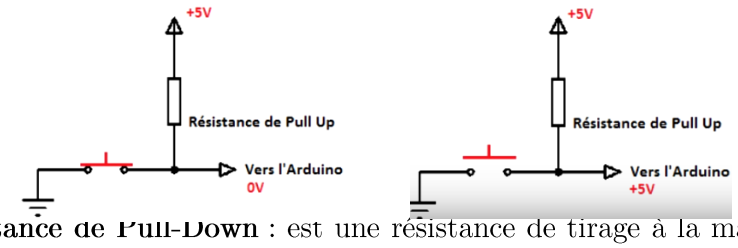
Exercice 2 : Interrupteur et Acquisition Numérique

Dans l'exercice 1 nous avons appris comment contrôler les ports d'un Arduino. Maintenant nous allons voir comment faire acquérir une information binaire (''0'' ou ''1'' logique) à un port Arduino.

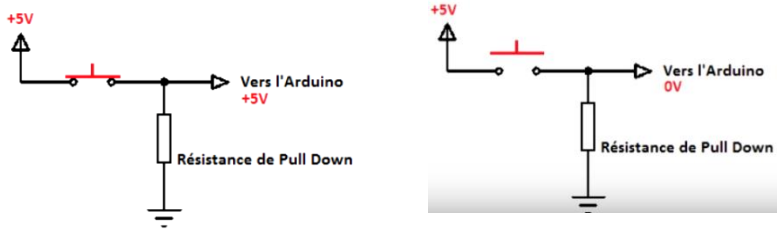
Ici nous allons utiliser un « bouton poussoir » comme étant un capteur d'état binaire (''0'' ou ''1'' logique).

Notes : Pour l'utilisation d'un bouton poussoir, il est impératif de comprendre la notion de « Résistance de Pull-Up » et « Résistance de Pull-Down ».

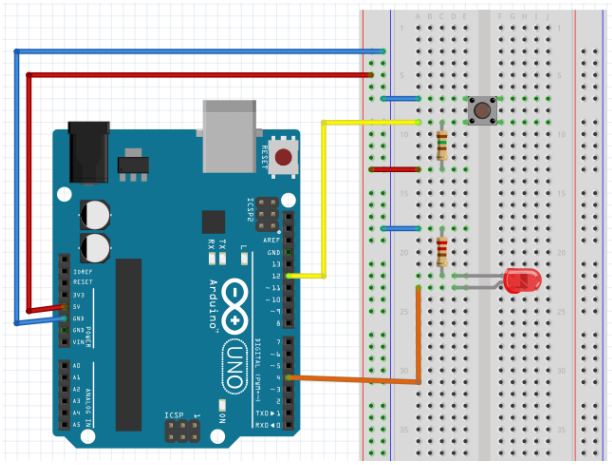
- **Résistance de Pull-Up** : permet de maintenir un état logique haut (5V) sur la broche Arduino.



- **Résistance de Pull-Down** : est une résistance de tirage à la masse, elle permet de maintenir un état bas (0V) sur la broche Arduino.

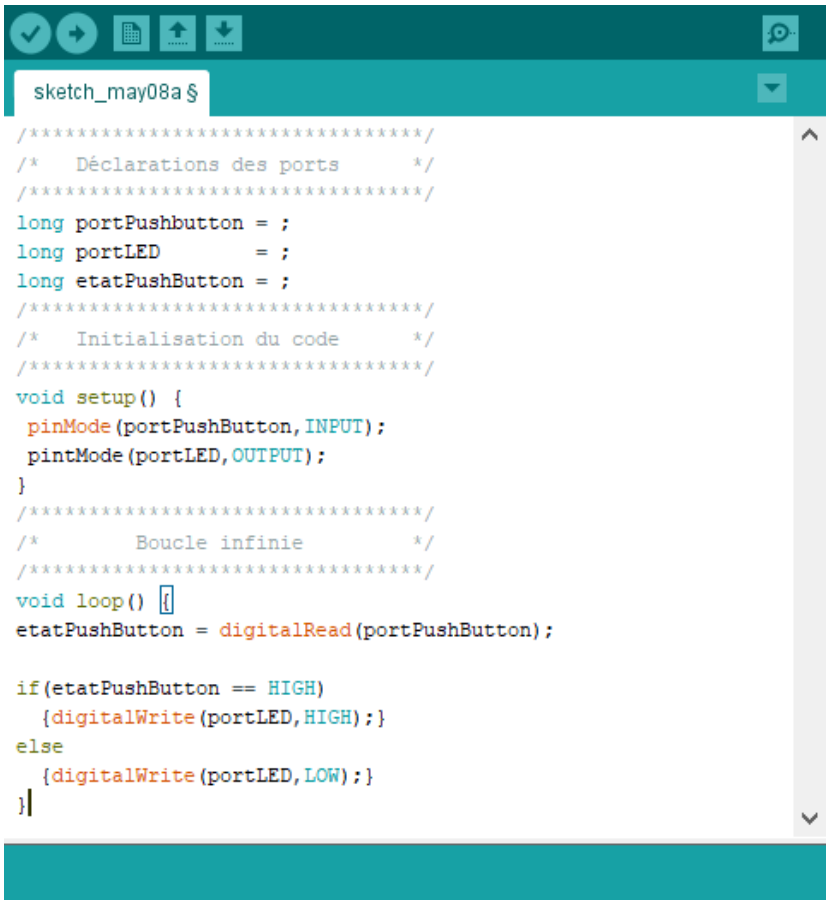


1. Réaliser le schéma de câblage (ci-dessous) en utilisant de logiciel Fritzing (platine d'essai, vue schématique et circuit imprimé), puis câbler le montage.



2. D'après le câblage, compléter le code programme ci-dessous.
   - portPushButton est l'affectation du port auquel est connecté le bouton poussoir
   - portLED : est le port auquel est connecté la LED

- etatPushButton : est l'état du bouton poussoir.

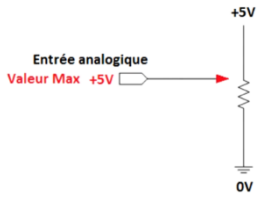


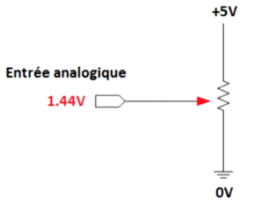
Remarque : Il se pourrait que la LED ne soit pas stable et cela s'explique simplement car il n'y a pas de résistance de Pull-Up. Pour cela, dans la fonction initialisation du programme, définissez le portPushButton comme une entrée Pull-Up avec la syntaxe suivante « **INPUT_PULLUP** ». Puis relancer le programme. Que constatez-vous ?

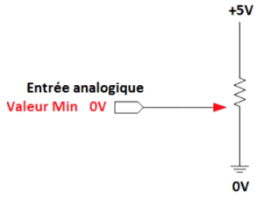
3. Il est possible de connaître la valeur lue par le bouton poussoir grâce au « Moniteur série » de l'IDE Arduino. Pour cela :
   - Compléter le code dans la partie initialisation juste au début de la fonction avec la syntaxe suivante « **Serial.begin.9600);** »
   - Puis dans la boucle infinie en dessous de la première instruction la syntaxe suivante « **Serial.println(etatPushButton);** ». Cette syntaxe permet de renseigner l'information nous souhaitons visualiser.
   - Téléverser le code puis lancer le moniteur série, puis actionner le bouton poussoir. Que constatez-vous ?
4. **BONUS** : Rajouter un deuxième bouton poussoir puis une LED sur le montage ensuite coder le programme de façon qu'un le deuxième bouton poussoir réalise la même fonction sur la deuxième LED2.

Exercice 3 : L'Acquisition Analogique avec un Potentiomètre

Pour faire l'acquisition analogique, nous allons utiliser un potentiomètre qui est un composant électronique qui permet de faire varier la tension. Dans le cas d'une tension d'alimentation de 5V, la sortie du potentiomètre peut varier de 0–5V. En effet, le composant fait un balayage de la plage de tension.

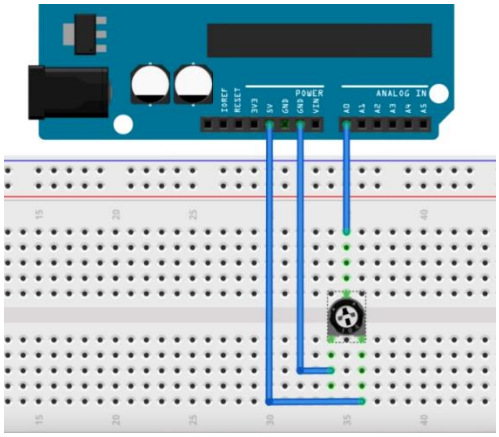






Position à 100% de la valeur max	Position à 28.8% de la valeur max	Position à 0% de la valeur max

1. Réaliser le schéma de câblage (ci-dessous) en utilisant de logiciel Fritzing (platine d'essai, vue schématique et circuit imprimé), puis câbler le montage.
   - La broche du signal sortant du potentiomètre doit être connectée sur une broche analogique de l'Arduino (A0–A5). Lors de l'affection des broches pendant la programmation, il n'est pas nécessaire de spécifier si la broche analogique utilisée est une Entrée ou Sortie car les broches analogiques (A0–A5) de l'Arduino sont toutes configurées en tant qu'entrées.



2. Programmer la carte en utilisant le code ci-dessous, puis exécuter le programme et lancer le moniteur série.


```
/**********************************/
/*   Déclarations des ports       */
/**********************************/
long portPotentiometre = A0 ; //affectation de la broche A0 au signal sortant du potentiomètre
long valeurPotentiometre; //décalaration de la valeur de la broche en tant que entrée Analog

/**********************************/
/*   Initialisation du code       */
/**********************************/
void setup() {
 Serial.begin(9600); //Initialisation du moniteur série pour afficher la valeur lue du potenti
}
/**********************************/
/*        Boucle infinie          */
/**********************************/
void loop() {
  valeurPotentiometre = analogRead(portPotentiometre);//On fait une lecteur sur l'entrée Analog
  Serial.println(valeurPotentiometre); //On demande d'afficher la valeur lue sur le moniteur
  delay(1); //délai d'attente pour la stabilité de la carte
```

Remarque : Le moniteur série vous affiche un nombre qui varie entre « 0–1023 » selon la position du potentiomètre. Les entrées analogiques de l'Arduino sont des convertisseurs Analogique/Numérique, et ces entrées A/N de l'Arduino numérise les entrées sur 10bits soit 2^{10}.

Ci-dessous la logique de conversion d'une donnée analogique en une donnée numérique :

Conversion Analogique/Numérique

Tension sur l'entrée Analogique		Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Valeur Numérisée	
		2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0		
		512	256	128	64	32	16	8	4	2	1		
Val Min	0V	0	0	0	0	0	0	0	0	0	0	0x512+0x256+0x128+0x64+0x32+0x16+0x8+0x4+0x2+0x1	0
Quantum	4.89mV	0	0	0	0	0	0	0	0	0	1	0x512+0x256+0x128+0x64+0x32+0x16+0x8+0x4+0x2+1x1	1
Exemple 1	1.67V	0	1	0	1	0	1	0	1	0	1	0x512+1x256+0x128+1x64+0x32+1x16+0x8+1x4+0x2+1x1	341
Exemple 2	2.5V	0	1	1	1	1	1	1	1	1	1	0x512+1x256+1x128+1x64+1x32+1x16+1x8+1x4+1x2+1x1	511
Exemple 3	3.3V	1	0	1	0	1	0	1	0	1	0	1x512+0x256+1x128+0x64+1x32+0x16+1x8+0x4+1x2+0x1	682
Val Max	5V	1	1	1	1	1	1	1	1	1	1	1x512+1x256+1x128+1x64+1x32+1x16+1x8+1x4+1x2+1x1	1023

En effet pour retrouver la valeur de tension, il faut convertir la valeur numérique (affichée sur le moniteur série) en une tension analogique :

$$f(x) = \frac{x \times 5}{1023}$$

Exemple : Pour la valeur numérique de 341 on a :

$$f(341) = \frac{341 \times 5}{1023} = 1.6667V$$

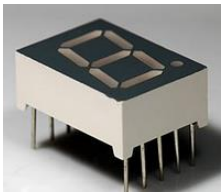
C'est la formule que nous allons intégrer dans notre programme afin de mettre en forme la valeur analogique du potentiomètre.

3. Mesurer la tension d'alimentation sur la broche (+) du potentiomètre (en mettant le potentiomètre sur sa position maximale) avec un multimètre ou la tension d'alimentation de l'Arduino. Puis en dessous de la première instruction de la boucle infinie taper la syntaxe en utilisant la formule de conversion N/A, sachant $f(x) = valeurPotentiomtre$ et $x = valeurPotentiometre$. Modifier le « **type de la variable valeurPotentiometre** » en « float » afin qu'elle puisse stocker des nombres à virgule. (**long valeurPotentiometre→float valeurPotentiometre**)
4. Téléverser le programme dans l'Arduino, puis lancer le moniteur série en actionnant le potentiomètre. Que constatez-vous ?

Exercice 4 : L'afficheur 7 segments

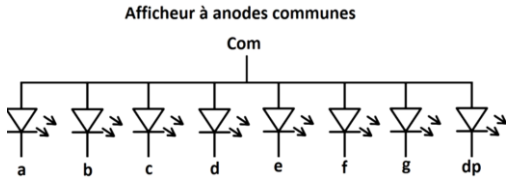
Définition Afficheur 7 segments :

Un afficheur 7 segments est un module électronique composé de 8 LED, dont 7 LED pour les segments et une 8ème LED pour le point (en bas à droite de l'afficheur).

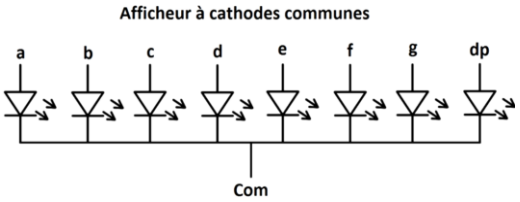


Il existe deux type d'afficheur 7 segments :

- Les afficheur à anodes communes : pour ce type d'afficheur, toutes les anodes sont connectées ensemble. Par conséquent, pour allumer une LED il est nécessaire d'imposer un état bas sur cette entrée.



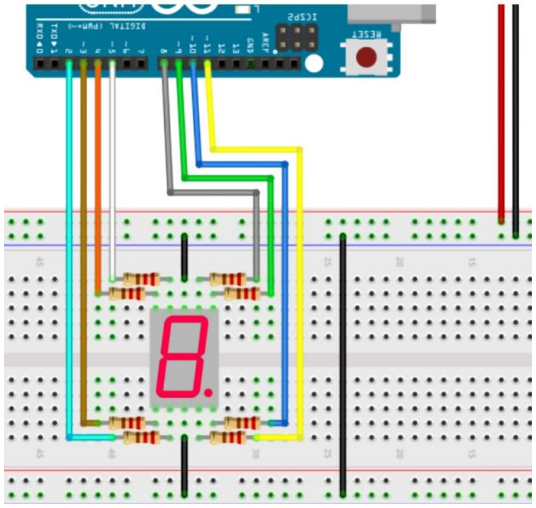
- Les afficheur à cathodes communes : pour ce type d'afficheur, toutes les cathodes sont connectées ensemble. Par conséquent, pour allumer une LED il est nécessaire d'imposer un état haut sur cette entrée.



Remarque : Pour savoir si l'afficheur est à anode commune ou cathode commune, il faut alimenter (5V) les LED de l'afficheur en laissant la borne (–) de l'alimentation fixe puis se balader avec la borne (+) et voir si toutes les LED s'allument. Si c'est le cas ce que l'afficheur est à anode commune. Inversement pour la cathode commune. Dans les deux cas, pensez à placer une résistance série (150Ω – 200Ω).

Dans le cas de cet exercice, nous utiliserons l'afficheur à cathode commune.

1. Réaliser le schéma de câblage (ci-dessous) en utilisant de logiciel Fritzing (platine d'essai, vue schématique et circuit imprimé), puis câbler le montage.

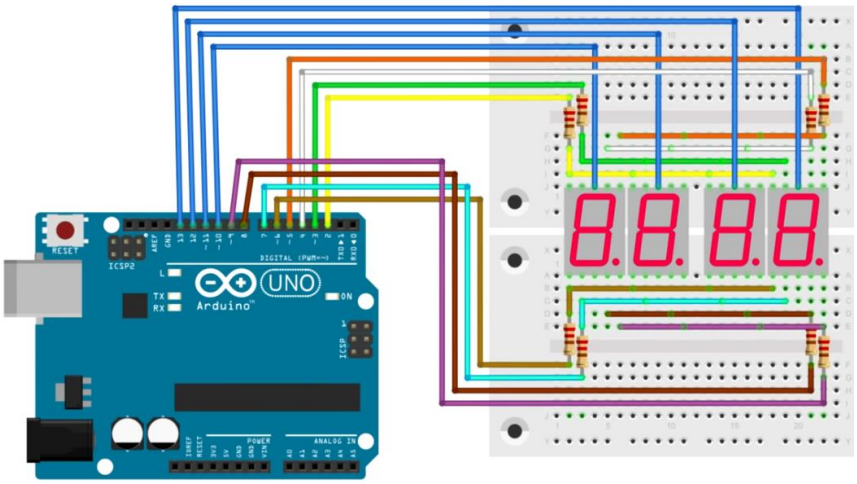


2. Programmer la carte en utilisant le code ci-dessous, puis exécuter le programme.
   - Attention, il y a une erreur entre le câblage et le code programme. Corriger l'erreur d'une des parties en fonction de l'autre.
   - Pour afficher le caractère « 0 », cela revient à allumer toutes les LED sauf les LED « DP » et « G » (voir les segments de l'afficheur). Ainsi pour afficher le caractère « 1 », cela revient à éteindre toutes les LED sauf les LED « B » et « C ».

sketch_may09a §

```
long ledPinA  = 2;//affectation de la broche/pin
long ledPinB  = 3;//affectation de la broche/pin
long ledPinC  = 4;//affectation de la broche/pin
long ledPinD  = 6;//affectation de la broche/pin
long ledPinE  = 7;//affectation de la broche/pin
long ledPinF  = 8;//affectation de la broche/pin
long ledPinG  = 9;//affectation de la broche/pin
long ledPinDP = 10;//affectation de la broch/pin
/**********************************/
/*   Initialisation du code       */
/**********************************/
void setup() {
pinMode(ledPinA, OUTPUT);//Initialisation de la broche/pin configurée en tant que sortie
pinMode(ledPinB, OUTPUT);//Initialisation de la broche/pin configurée en tant que sortie
pinMode(ledPinC, OUTPUT);//Initialisation de la broche/pin configurée en tant que sortie
pinMode(ledPinD, OUTPUT);//Initialisation de la broche/pin configurée en tant que sortie
pinMode(ledPinE, OUTPUT);//Initialisation de la broche/pin configurée en tant que sortie
pinMode(ledPinF, OUTPUT);//Initialisation de la broche/pin configurée en tant que sortie
pinMode(ledPinG, OUTPUT);//Initialisation de la broche/pin configurée en tant que sortie
pinMode(ledPinDP, OUTPUT);//Initialisation de la broche/pin configurée en tant que sortie
}
/**********************************/
/*        Boucle infinie          */
/**********************************/
void loop() {
afficher0();//On demande d'afficher le caractère '0' en exécutant la fonction 'Afficher 0'
delay(1000);//On demande d'afficher le caractère '0' pendant 1 seconde
afficher1();//On demande d'afficher le caractère '1' en exécutant la fonction 'Afficher 1'
delay(1000); //On demande d'afficher le caractère '1' pendant 1 seconde
}
/**********************************/
/*        Fonction Afficher 0     */
/**********************************/
void afficher0(){
  digitalWrite(ledPinA,HIGH);//On allume le segment A
  digitalWrite(ledPinB,HIGH);//On allume le segment B
  digitalWrite(ledPinC,HIGH);//On allume le segment C
  digitalWrite(ledPinD,HIGH);//On allume le segment D
  digitalWrite(ledPinE,HIGH);//On allume le segment E
  digitalWrite(ledPinF,HIGH);//On allume le segment F
  digitalWrite(ledPinG,LOW);//On éteint le segment G
  digitalWrite(ledPinDP,LOW);//On éteint le segment DP
}
```

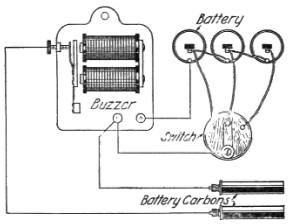
3. Compléter le code programme pour réaliser les autres fonctions qui permettent d'afficher les caractères restants (1, 2, 3, 4, 5, 6, 7, 8, 9).
4. Faites le câblage de l'afficheur qui permet d'afficher 4 caractères en même temps puis réduire le temps d'intervalle de chacune des fonctions afin qu'on ait un affichage continu.



Exercice 5 : Le Buzzer

Définition le Buzzer :

Le Buzzer est un composant électromagnétique ou piézoélectrique qui transforme l'énergie électrique en vibration acoustique. Il produit un son caractéristique « le bip » quand on lui applique une tension.

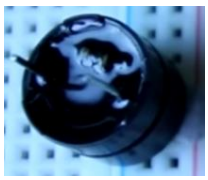


Il existe deux types de Buzzer :

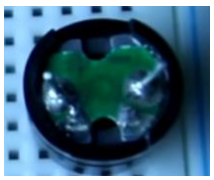
- Buzzer Actif : le buzzer actif peut fonctionner avec une tension continue
- Buzzer Passif : le buzzer actif fonctionne uniquement avec une tension alternative dont la fréquence est généralement comprise entre 500Hz–5kHz

Notes : Pour faire la différence entre un buzzer actif et un passif : le buzzer actif est complètement encapsulé et sur le buzzer passif le PCB « Printed Circuit Board », en français « Circuit Imprimé » est visible.

<u>Remarque</u> : Le Buzzer n'est pas un haut-parleur.



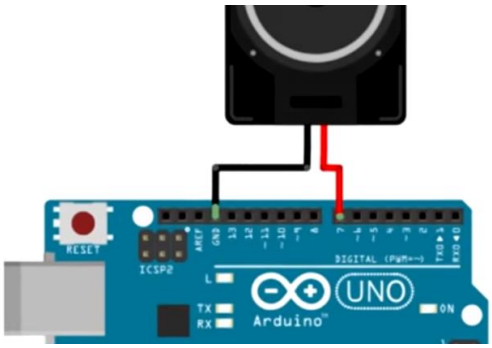
Actif



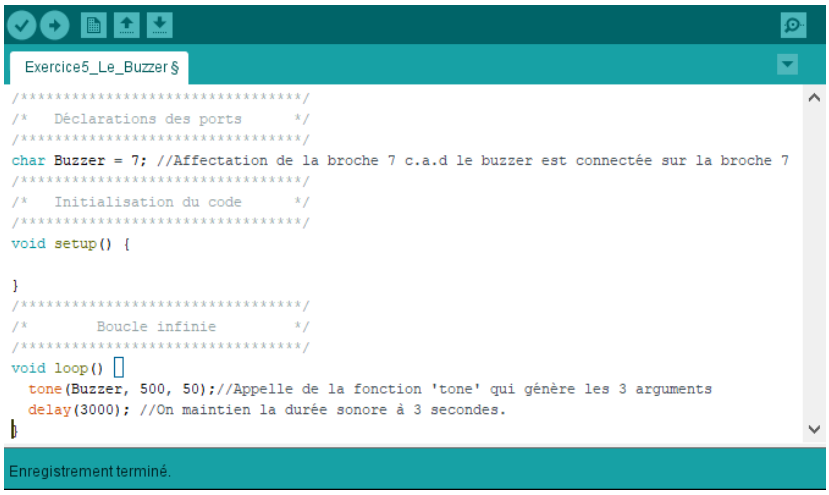
Passif

Le Buzzer permet d'obtenir une information sonore contrairement à la LED qui nous donne une information visuelle. Il peut utiliser pour signaler l'anomalie de fonctionnement d'un système ou de créer une alarme.

1. Réaliser le schéma de câblage (ci-dessous) en utilisant de logiciel Fritzing (platine d'essai, vue schématique et circuit imprimé), puis câbler le montage.
   - **Attention** à ne pas connecter le Buzzer sur la broche/pin « **3** » et « **11** », car ces deux broches servent pour le PWM et cela peut interférer avec l'usage du buzzer.
   -



2. Programmer la carte en utilisant le code ci-dessous, puis exécuter le programme.
   - Pour créer une notification il suffit d'appeler la fonction « tone ». Cette fonction permet de générer un signal carré selon 3 arguments.
     - L'argument 1) La broche numérique où est connectée le buzzer.
     - L'argument 2) La fréquence du signal carré.
     - L'argument 3) La durée du signal.

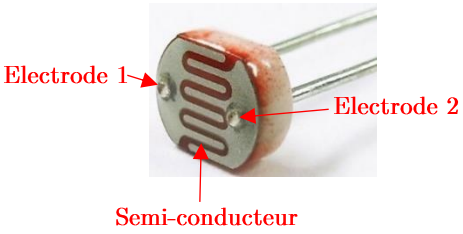


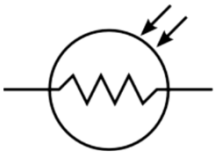
3. Connecter une LED sur les deux broches où est connecté le buzzer puis observer le résultat (Tenez à bien respecter la polarisation de la LED lors du branchement).
4. Programmer à nouveau la carte en utilisant le code ci-dessous.
   - Veillez commenter chacune des lignes du code.
   - Expliquez le raisonnement de ce code programme.
   - Quelle est la fréquence maximale de ce signal ?
   - Quel est la durée du signal ?
   - Quelle est la fonction qui réalise les sons graves aux aigus et puis celle qui réalise les sons aigus aux graves ?

Exercice 6 : Le Capteur de luminosité « La Photorésistance »

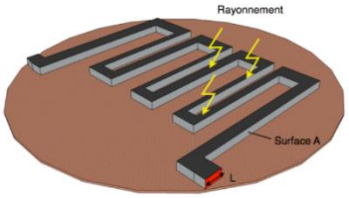
Définition la Photorésistance :

La photorésistance est un semi-conducteur dont la résistance évolue en fonction de son exposition à la lumière. Elle est également appelée « **Résistance Photo-dépendante** » ou « **Cellule Photoconductrice** ».

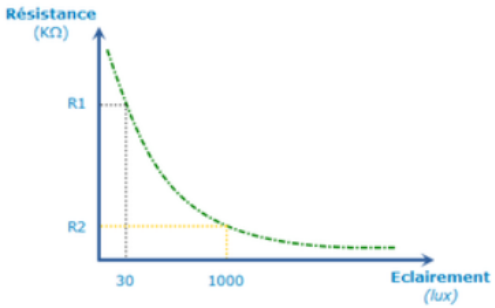




Symbol photorésistance

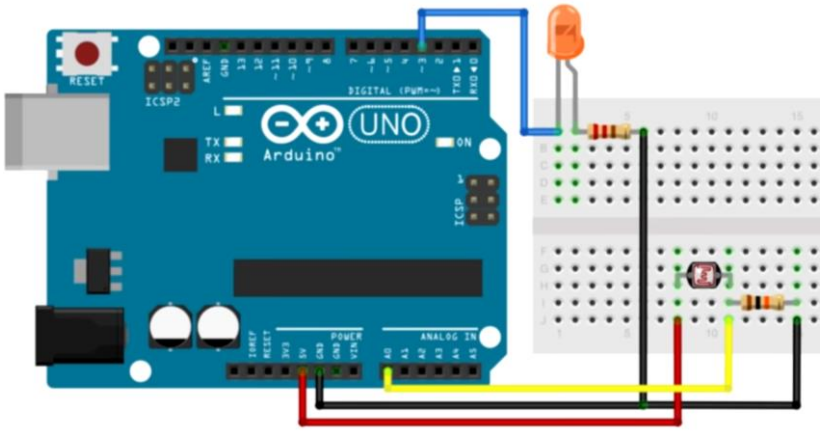


La photorésistance est composée de deux électrodes séparées par un semi-conducteur en forme de « zig-zag ». Lorsque ce semi-conducteur va absorber les photons qui constitue la lumière, ces mêmes photons vont fournir de l'énergie aux électrons de façon à facilité leurs déplacements entre les deux électrodes. En effet, si le déplacement **des électrons est facilité alors la résistance de la photorésistance diminue.**



<u>Notes</u> : Plus la photorésistance est exposée à la lumière plus sa résistance sera faible. Sa forme en « zig-zag » permet d'optimiser sa sensibilité à la lumière.

1. Réaliser le schéma de câblage (ci-dessous) en utilisant de logiciel Fritzing (platine d'essai, vue schématique et circuit imprimé), puis câbler le montage.
   - La résistance placée entre A0 et la masse vaut 10kΩ et celle de la LED vaut 220Ω.

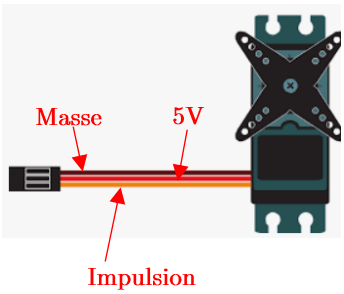


2. Programmer la carte en utilisant le code ci-dessous, puis exécuter le programme.
   - Rajouter les commentaires manquants sur certaines instructions.
   - La valeur de l'OFFSET dépendra de votre environnement. N'hésitez pas de la modifier.

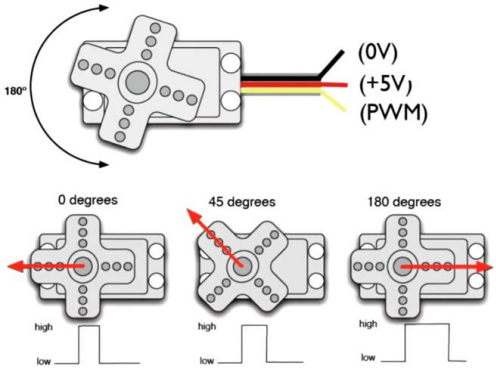
Exercice 7 : Le Servomoteur

Définition Servomoteur :

Le Servomoteur est un moteur à courant continu piloté par un driveur qui régule constamment sa position. Un servomoteur est un moteur capable de maintenir une position à un effort statique et dont la position est vérifiée en continu et corrigé en fonction de la mesure. C'est donc un système asservi. Un servomoteur comporte également un « **Réducteur** » qui sert à augmenter leur couple.



Pour piloter un servomoteur, on utilise la PWM (en français MLI pour Modulation de Largueur d'Impulsions), car c'est « **le rapport cyclique** » de la PWM qui détermine l'angle de positionnement.



<u>Notes</u> : Un servomoteur est limité à un débattement de 180°, il est donc pas conseillé de faire faire un tour complet (360°) à un servomoteur.

- **La MLI** : la Modulation de Largeur d'Impulsions consiste à faire varier le rapport cyclique d'un signal de période fixe.
- **Le Rapport cyclique** : en électronique désigne phénomène périodique, **le rapport** de la durée du phénomène sur une période et la durée de cette même période.

<u>Remarque</u> : Pour l'utilisation d'un servomoteur avec le module Arduino, le servomoteur doit être câblé sur une sortie PWM.

1. Programmer la carte en utilisant le code exemple pour le pilotage d'un servomoteur, et pour cela aller dans : **File→Exemples→Servo→Sweep**.
2. D'après le code programme, câbler correctement le servomoteur sur la carte Arduino en respectant le brochage (Masse, 5V et Impulsion).
3. Les commentaires du code programme seront normalement en anglais, essayer de les comprendre puis traduire directement les commentaires sur le code.

Nous allons maintenant commander le servomoteur en temps réel en utilisant un potentiomètre (rappelez-vous des notions et exercice sur le potentiomètre).

4. Commande en temps réel du servomoteur :
   - A) Connecter le potentiomètre sur une entrée Analog de la carte Arduino. Veillez respecter le brochage du potentiomètre.
   - B) Programmer la carte en utilisant le code ci-dessous
     - **ATTETION** : le code est incomplet, veillez le compléter, par la même occasion veillez remplacer la première instruction de la boucle principale par sa variable déclarée plus haut dans le code.

<u>Notes</u> : La fonction « map » permet de faire le produit en croix entre la valeur numérique du potentiomètre (1023) et l'angle de rotation du servomoteur (0 – 180°).

sketch_may11a §

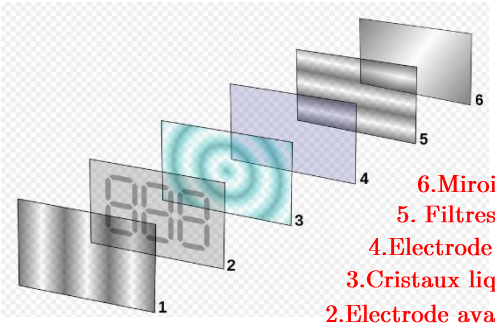
```
#include <Servo.h>

Servo monServo;
char PWM_SERVO = 9;
char POTENTIOMETRE = 0;
int positionDuServo = 0;

void setup() {
  monServo.attach(9);
  monServo.write(positionDuServo);

}

void loop() {
 positionDuServo = analogRead(POTENTIOMETRE);
 positionDuServo = map(val, 0, 1023, 0, 180);
 BLABLABLBABLABLABLABLABLABLABLABLABLABLABL;
 HOHOHOHOHOHOOHOHOHOHOOHOHOHOHOHOOHOHOHOHOOHOHOHO.
}
```

5. Commenter votre code, puis téléverser le programme sur la carte Arduino. Actionner le potentiomètre puis observer le résultat. Que constatez-vous ?

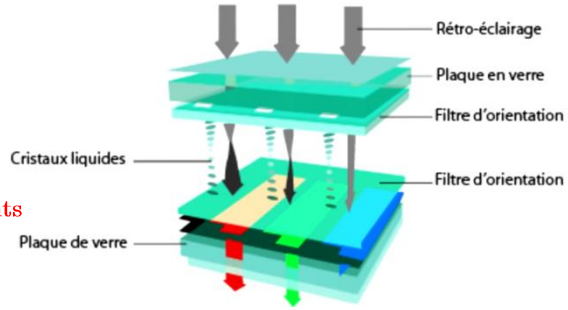
Exercice 8 : L'Afficheur LCD

Définition Afficheur LCD :

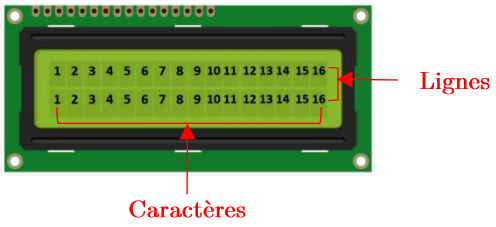
L'Afficheur LCD « **Liquid Crystal Display** en anglais » et « en français **Ecran à Cristaux Liquides** » utilisent une polarisation de la lumière par des filtres. En résumé, la lumière est filtrée par des cristaux liquides car c'est le rétro-éclairage à l'arrière de l'afficheur qui sert de source lumineuse (voir images ci-dessous).



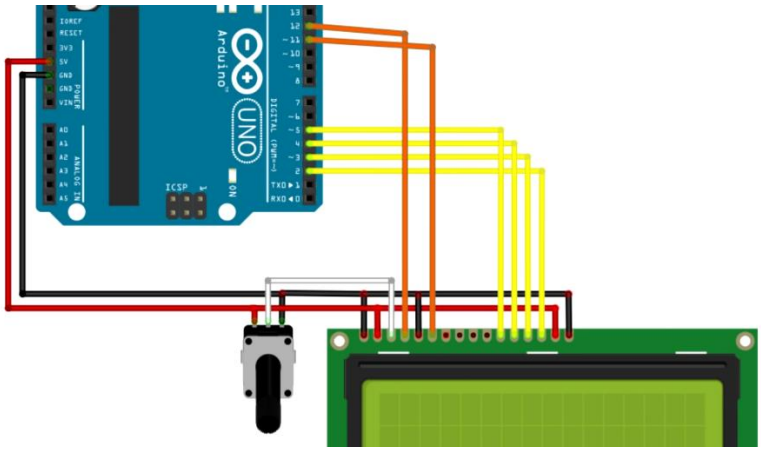
6.Miroir
5. Filtres polarisants
4.Electrode arrière
3.Cristaux liquides
2.Electrode avant
1.Filtres polarisants



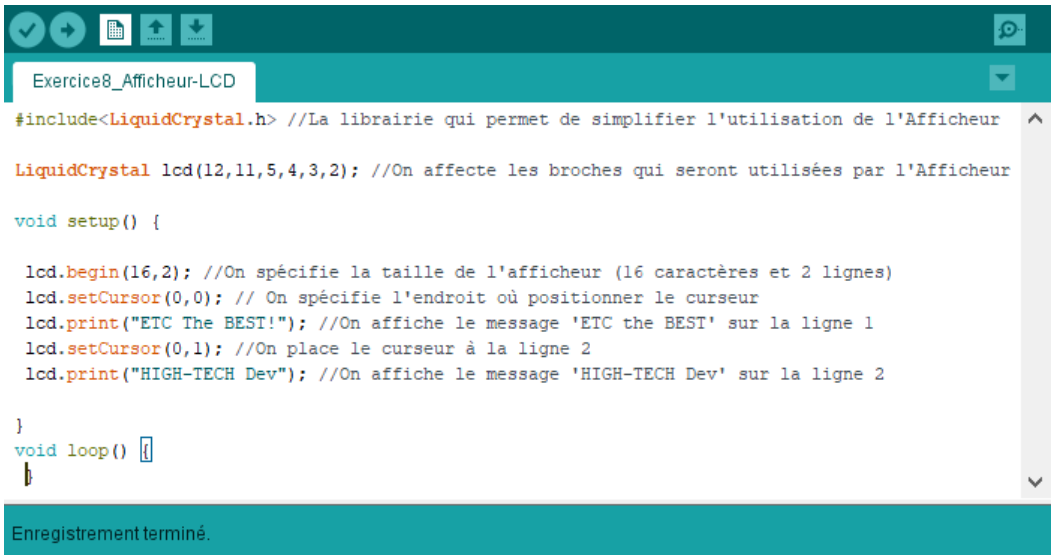
Dans le cadre de cet exercice, nous allons utiliser l'afficheur de référence « **1602A** » qui signifie « **2 lignes de 16 caractères** ».



1. Réaliser le schéma de câblage (ci-dessous) en utilisant de logiciel Fritzing (platine d'essai, vue schématique et circuit imprimé), puis câbler le montage.
   - Le **potentiomètre** permet de **régler le contraste de l'écran.**



2. Programmer la carte en utilisant le code ci-dessous, puis exécuter le programme.
   - La librairie « LiquidCrystal » permet de simplifier l'utilisation de l'afficheur.
   - Le curseur indique à partir d'où le message sera affiché.



3. Pour faire clignoter votre message, utiliser les instructions ci-dessous dans la boucle infinie :

```
lcd.noDisplay(); //On éteint l'Afficheur
delay(500); //....pendant une durée de 500ms
lcd.display(); //On rallume l'Afficheur
delay(500);// ....pendant une durée de 500ms
```

4. Pour faire dérouler votre message sur la gauche, utiliser les instructions ci-dessous dans la boucle infinie :
   - Pour que cette fonction marche, veillez placer le curseur à droite

```
lcd.scrollDisplayLeft();
delay(300);
```

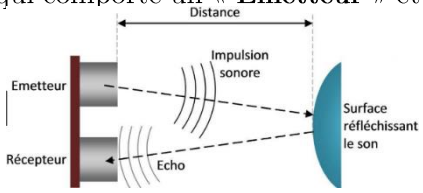
5. Réaliser la fonction inverse de la question précédente.
6. **BONUS** : Il est possible également possible de taper votre message directement depuis le Serial Moniteur de l'IDE Arduino. Pour cela utiliser la méthode suivante :
   - Une fois la carte programmée, lancer le moniteur série pour afficher votre message.

Exercice 9 : Un Capteur à Ultrason

Définition Capteur à Ultrason :

Un capteur à Ultrason est un transducteur qui comporte un « **Emetteur** » et un « **Récepteur** » d'ondes.

HC–SR04



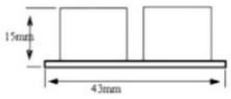
Lorsque le récepteur reçoit un « **écho** » de l'onde émise par l'émetteur, il est en effet, possible en fonction de l'intervalle de temps (la distance) et connaissant la vitesse de la propagation du son de déterminer la distance qui sépare avec l'obstacle. En effet, plus le délai est court, plus l'obstacle est proche, inversement.

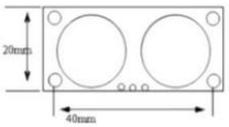
Dans le cadre de nos différents projets au sein de ETC, nous utiliserons le capteur à Ultrason de référence « **HC–SR04** ». Ci-dessous les caractéristiques de ce capteur à ultrason

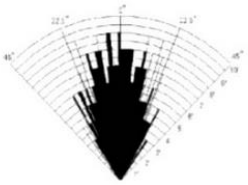
Features:

- Power Supply :+5V DC
- Quiescent Current : <2mA
- Working Currnt: 15mA
- Effectual Angle: <15°
- Ranging Distance : 2cm – 400 cm/1" - 13ft
- Resolution : 0.3 cm
- Measuring Angle: 30 degree
- Trigger Input Pulse width: 10uS
- Dimension: 45mm x 20mm x 15mm

VCC = +5VDC
Trig = Trigger input of Sensor
Echo = Echo output of Sensor
GND = GND



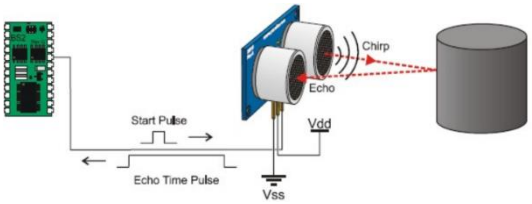


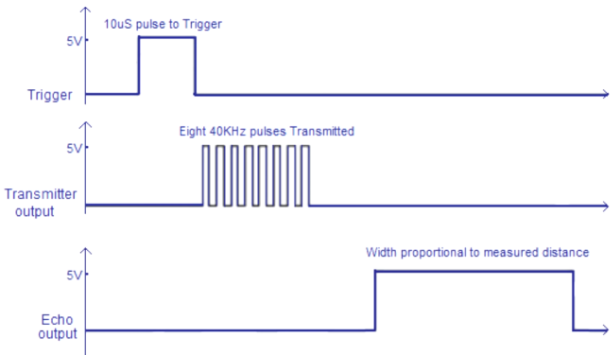


Practical test of performance, Best in 30 degree angle

<u>Fonctionnement du capteur à Ultrason **HC–SR04**</u> :

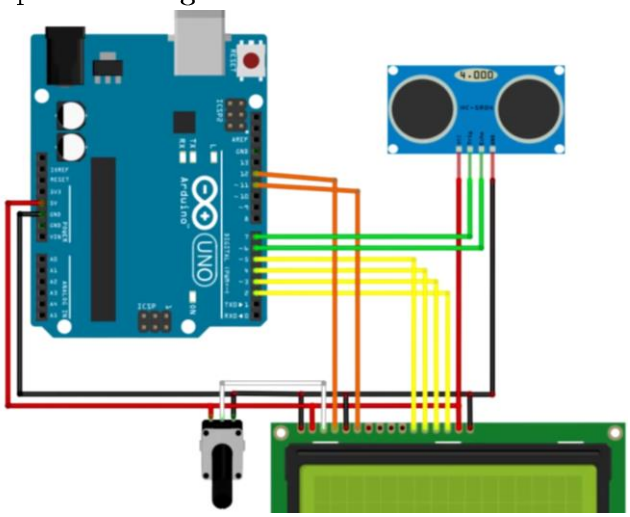
Lorsque le capteur reçoit une impulsion 5V de 10µs (10 microsecondes) sur son entrée « Trigger », il va émettre des **ultrasons** à la fréquence de **40kHz** et lorsque le récepteur reçoit ces mêmes ultrasons, la sortie écho envoie une impulsion 5V dont la durée est proportionnelle à la distance selon le rapport suivant : $Distance\ (cm) = \frac{Durée\ de\ l'impusion(\mu s)}{58{,}8}$





Pour cet exercice, nous allons mettre en place un système de détection de la distance d'un objet et d'afficher cette information sur un écran LCD.

1. Réaliser le schéma de câblage (ci-dessous) en utilisant de logiciel Fritzing (platine d'essai, vue schématique et circuit imprimé), puis câbler le montage.
   - Le **potentiomètre** permet de **régler le contraste de l'écran.**



   - L'**afficheur LCD** permet d'**afficher la distance mesurer.**
2. Programmer la carte en utilisant le code ci-dessous, puis exécuter le programme.
   - Le code ci-dessous est incomplet, veillez le compléter. Rappelez-vous des notions concernant utilisation d'un afficheur LCD.
   - D'après les explications sur le fonctionnement du capteur à ultrason, commenter le programme en expliquant le raisonnement et fonctionnement.

Exercice9_Capteur-Ultrason §

```
LiquidCrystal lcd(12,11,5,4,3,2);
char DOUT_TRIGGER = 7;
char DIN_ECHO     = 6;
float distance;

void setup() {
  lcd.begin(16,2);
  pinMode(DOUT_TRIGGER, OUTPUT);
  pinMode(DIN_ECHO, INPUT);
}

void loop() {
  digitalWrite(DOUT_TRIGGER, LOW);
  delayMicroseconds(2);
  digitalWrite(DOUT_TRIGGER, HIGH);
  delayMicroseconds(10);
  digitalWrite(DOUT_TRIGGER, LOW);

  distance = pulseIn(BLABLABLABLABLA, HIGH)/58.0;

  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Distance");
  lcd.setCursor(0,1);
  lcd.print(distance);
  lcd.print(" cm");
  delay(200);
```