TP N°1 : Un feu tricolore

Partie 3 : « L'énergie et ses conversions »

Référentiel :

« Réaliser des circuits simples et exploiter les lois de l'électricité. »

Usages du micro-contrôleur :

- Réalisation d'un générateur de tension continue dont la nature des pôles change au cours du temps.

Activités de programmation réalisées par les élèves :

- Modification du code initial pour concevoir un feu de signalisation tricolore.
(Ce premier TP, proposé à des élèves de quatrième, permet d'introduire les principales fonctions du langage Arduino™)

TP N°1 : Un feu tricolore		
NOM : Prénom : *Classe* :		Durée : 60 minutes
Objectif : - Fixer la position des pôles d'un générateur de tension en programmant un micro-contrôleur - Programmer un feu de signalisation tricolore		
Compétences évaluées	**Domaine**	Evaluation
1. Concevoir une expérience pour la tester.	D4	
2. Utiliser des outils d'acquisition et de traitement de données, de simulations et de modèles numériques.	D2	

I. Buts du TP

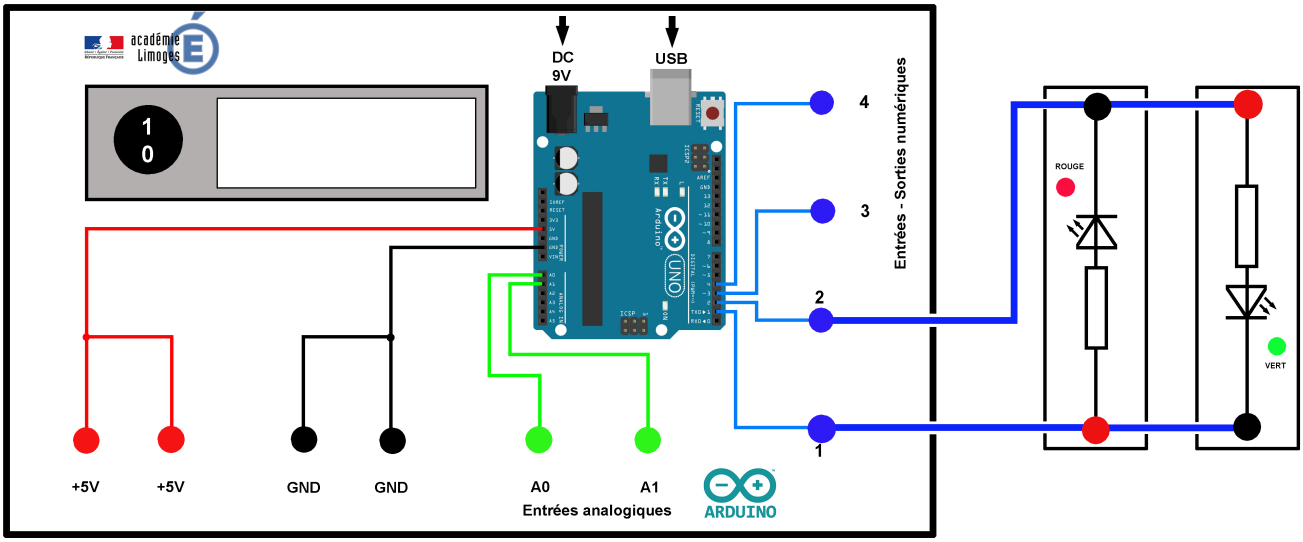
*On peut utiliser **deux sorties numériques d'un micro-contrôleur Arduino™ pour créer un générateur de tension continue**. Un programme simple permet de générer une tension de 5 V entre ces deux sorties et de **fixer la position des bornes + et – du générateur** réalisé.*

*En reliant des DEL à ces sorties, on vérifiera dans un premier temps que **le courant électrique circule bien de la borne + vers la borne –** à l'extérieur du générateur.*

*On programmera ensuite, en modifiant le code initial, le fonctionnement d'un **feu de signalisation tricolore**.*

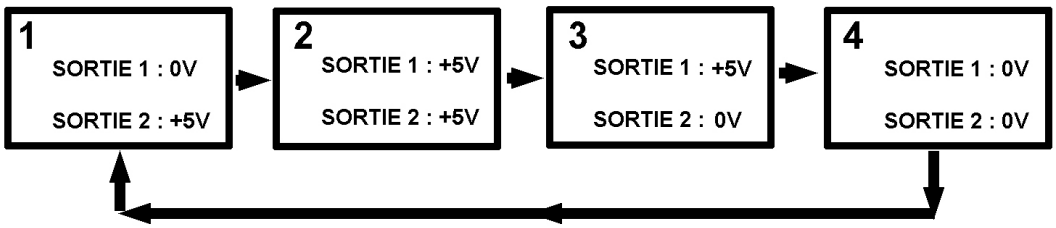
II. Etude préparatoire

– **Réaliser le montage schématisé ci-dessous :**



– **Relier le PC à la platine à l'aide du câble USB.**
– **Ouvrir le programme *blink.ino* .**
– **Téléverser ce programme vers l'Arduino™ .**

Le programme ***blink.ino*** *permet de générer alternativement sur les sorties 1 et 2 les* ***potentiels électriques 0V ou +5V****. Toutes les 10 secondes, les valeurs de ces potentiels électriques sont modifiées selon la séquence suivante :*



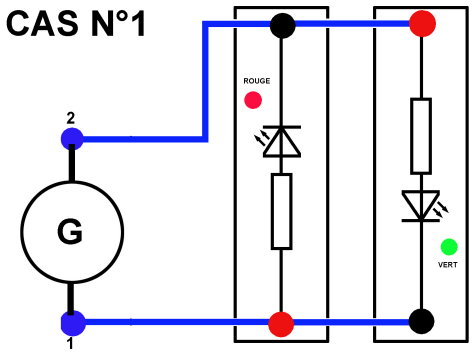
1. **Dans quels cas voit-on une DEL s'allumer ? Dans quels cas toutes les DEL sont-elles éteintes ?**

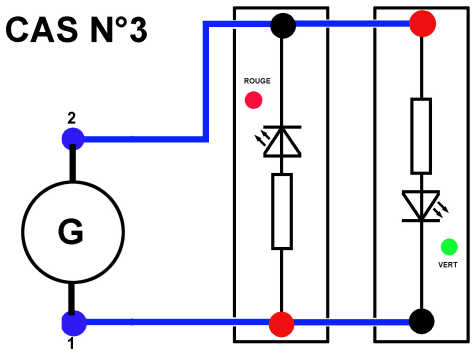
---

2. **En déduire une condition pour qu'une tension non nulle existe entre deux sorties numériques.**

---

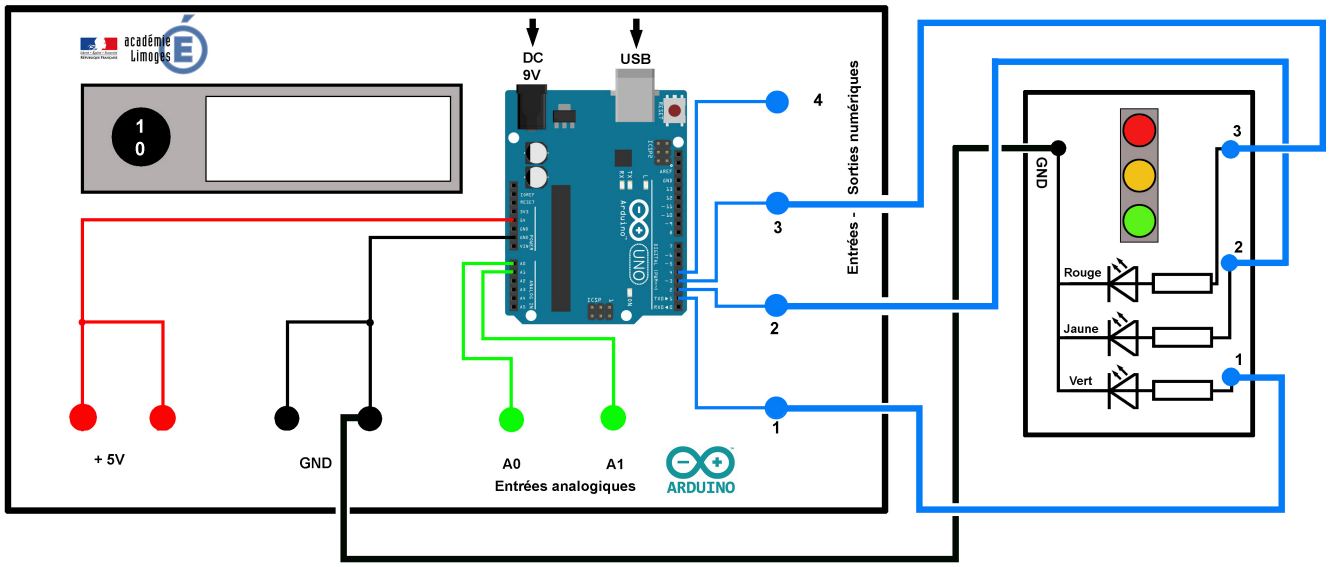
3. **Compléter les schémas équivalents pour les cas 1 et 3 en indiquant la position des bornes + et – , le sens du courant dans le circuit et la DEL allumée.**





III. Programmation d'un feu tricolore

1. Réaliser le montage schématisé ci-dessous :



2. Modification du code initial – programmation du feu de signalisation

*Le code du programme **blink.ino** a été reproduit ci-dessous. Les explications des lignes de code sont indiquées en italique. Après avoir lu attentivement ce code et compris son architecture, modifier les parties écrites en gras dans les cadres pour commander l'allumage des DEL du feu tricolore et reproduire le fonctionnement du montage présenté par le professeur.*

```
// Appel des bibliothèques prenant en charge l'écran
  #include <Wire.h>
  #include <LiquidCrystal_I2C.h>
// Calibration de l'écran
  LiquidCrystal_I2C lcd(0x27, 16, 2);
```

```
// Définition des bornes de la platine qui seront utilisées.
// La variable borne1 est un entier ( fonction int ). A COMPLÉTER

  int borne1 = 1 ;
  int borne2 = 2 ;
```

// initialisation du programme. Cette partie du programme n'est lue qu'une seule fois

```
void setup() {
```

// Les bornes 1 et 2 sont définies comme des sorties (OUTPUT). À COMPLETER

```
  pinMode(borne1, OUTPUT);
  pinMode(borne2, OUTPUT);
```

// initialisation de l'écran

```
   lcd.init();
```

// Allumage du rétroéclairage de l'écran

```
   lcd.backlight();
```

// Ecriture d'un message permanent sur les deux premières lignes : lcd.setCursor(0,0) permet de positionner le message au niveau du pixel 0 de la ligne 0. lcd.setCursor(0,1) permet de positionner le message au niveau du pixel 0 de la ligne 1. À SUPPRIMER !

```
   lcd.setCursor(0,0) ; lcd.print(" SORTIE 1 :") ;
   lcd.setCursor(0,1) ; lcd.print(" SORTIE 2 :") ;
```

```
  }
```

// la suite du programme tourne en boucle et sera lue indéfiniment

```
void loop() {
```

// TOUTE CETTE PARTIE DU PROGRAMME DOIT ÊTRE MODIFÉE

// Ecriture des valeurs 0V sur la sortie 1, +5V sur la sortie 2. 0V correspond à un niveau bas (LOW) , 5V à un niveau haut (HIGH)

```
   digitalWrite(borne1, LOW);
   digitalWrite(borne2, HIGH);
```

// Affichage sur l'écran des potentiels électriques correspondants

```
   lcd.setCursor(11,0) ; lcd.print(" 0V") ;
   lcd.setCursor(11,1) ; lcd.print("+5V") ;
```

// attente d'une durée de 10 secondes soit 10000 millisecondes

```
 delay(10000);
```

```
// Ecriture des valeurs +5V sur la sortie 1, +5V sur la sortie 2
   digitalWrite(borne1, HIGH);
   digitalWrite(borne2, HIGH);

// Affichage sur l'écran des potentiels électriques correspondants
   lcd.setCursor(11,0) ; lcd.print("+5V") ;
   lcd.setCursor(11,1) ; lcd.print("+5V") ;

// attente d'une durée de 10 secondes soit 10000 millisecondes
   delay(10000);

// Ecriture des valeurs +5V sur la sortie 1, 0V sur la sortie 2
   digitalWrite(borne1, HIGH);
   digitalWrite(borne2, LOW);

// Affichage sur l'écran des potentiels électriques correspondants
   lcd.setCursor(11,0) ; lcd.print("+5V") ;
   lcd.setCursor(11,1) ; lcd.print(" 0V") ;

// attente d'une durée de 10 secondes soit 10000 millisecondes
   delay(10000);

// Ecriture des valeurs 0V sur la sortie 1, 0V sur la sortie 2
   digitalWrite(borne1, LOW);
   digitalWrite(borne2, LOW);

// Affichage sur l'écran des potentiels électriques correspondants
   lcd.setCursor(11,0) ; lcd.print(" 0V") ;
   lcd.setCursor(11,1) ; lcd.print(" 0V") ;

 // attente d'une durée de 10 secondes soit 10000 millisecondes
   delay(10000);  }
```

PHOTOS DU MONTAGE