Arduino PR 05

Feu Tricolore

Base

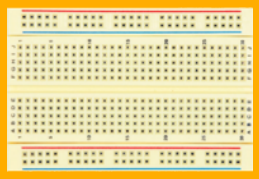
Nous aurons besoin de

L'Arduino Uno

5 LED s (2 vert , 2rouge, 1 jaune)

Le cable USB

L'ordinateur



La planche d'essai

2 résistance 220 Ω

Le programme Feu Tricolore

L'éditeur de programme

Le but est de reproduire les feux de circulation avec 5 LEDs :

- t1 : Les voitures passent
    - Côté piétons, le rouge est allumé, le vert éteint
    - Côté voitures, le vert est allumé, les autres éteints
- on attend 3s pour passer à t2
- t2 : Orange pour les voitures
    - Côté piétons, le rouge reste allumé, le vert éteint
    - Côté voiture, le orange s'allume, et les autres sont éteints
- on attend 1s pour passer à t3
- t3 : Les piétons passent
    - Côté piétons, le vert est allumé, le rouge éteint
    - Côté voiture, le rouge est allumé.
- on attend 3s pour passer à t1

Dans un premier temps, on place les LEDS sur le circuit et on vérifie qu'elles marchent avec un programme qui les allume toutes.

Dans la fonction `setup` il faudra déclarer tous les pins correspondant aux LEDs. Comme il y en a beaucoup on va les déclarer en série et utiliser une boucle

```
int PIETON_ROUGE = 13;
int PIETON_VERT = 12;
int VOITURE_VERT = 11;
int VOITURE_ORANGE = 10;
int VOITURE_ROUGE = 9;
int DUREE = 3000; // 3 secondes

// Cette fonction est utilisee quand on demarre l'Arduino
void setup() {
  // initialize digital pin 13 as an output.
  for (int l=9; l<=13; l++) {
    pinMode(l, OUTPUT);
  }
}
```

On peut de la même manière, utiliser une boucle pout allumer toutes les LEDs en remplace `pinMode` par `digitalWrite`.

Maintenant, on va faire clignoter toutes les LEDs.

Dans la fonction `loop`, on va utiliser l'heure pour allumer et éteindre. La fonction `loop` est appelée très régulièrement par l'Arduino. A chaque fois, on regardera l'heure et s'il s'est écoulé plus qu'une 1/2 seconde, on inverse l'état des LEDs.

```
void loop() {
  unsigned long maintenant = millis();
  if (maintenant - dernierTemps >= interval) {
    // releve le temps pour le prochain passage
    dernierTemps = maintenant;
    // retourne l'etat. si c'est allume, on eteint et
vice-versa
    …
}
```

La fonction `millis()` donne l'heure sous la forme d'un très long nombre. C'est le nombre de millisecondes écoulées depuis le 1er Janvier 1970.

Pour inverser l'état des LEDs, on va conserver l'état dans une variable, tester la valeur et mettre à HIGH si la valeur est LOW et à LOW si la valeur est HIGH.

```
  // si c'est allume, on eteint et vice-versa
    if (etat == LOW) {
      etat = HIGH;
    } else {
      etat = LOW;
    }
```

Ensuite la valeur est écrite avec `digitalWrite` dans une boucle `for` comme pour l'initialisation des pins.

Maintenant, on va changer le programme pour que l'heure nous permette de trouver dans quelle phase on est.

On va associer chaque seconde de 0 à 6 à chacune des phases et les reporter dans un tableau.

```
const byte VOITURE = 1;
const byte TRANSITION = 2;
const byte PIETON = 3;
const byte phases[] =  { VOITURE, VOITURE, VOITURE,
                         TRANSITION, PIETON, PIETON, PIETON };
```

On va calculer le nombre de secondes en divisant par les ms par 1000, puis prendre le modulo 7, et l'arrondir à un nombre entier. Le modulo est le reste de la division par 7 et donc on aura un nombre entre 0 et 6.

```
int t = round((maintenant - dernierTemps) / 1000 % 7);
int phase =  phases[t];
```

Une fois que l'on connait la phase, on pourra allumer les LEDs utilisées dans cette phase.

On commence par tout arrêter avec `changeEtatTout(LOW);`

Le bloc `switch/case` permet de traiter chacune des 3 phases.

```
switch (phase) {
    case VOITURE:
        digitalWrite(VOITURE_VERT, HIGH);
        digitalWrite(PIETON_ROUGE, HIGH);
        break;
    case PIETON:
        …
        break;
}
```