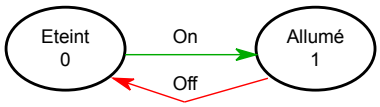
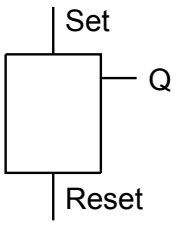
TP : Graphes et machines d'état

L'objectif de ce TP est de programmer des machines d'état avec un microcontrôleur.

1) Bascule Set-Reset

La bascule Set-Reset est probablement la machine d'état la plus simple qu'on puisse imaginer. Elle a deux états. Voici son graphe d'état et son symbole électronique :





On peut facilement la réaliser par le programme suivant :

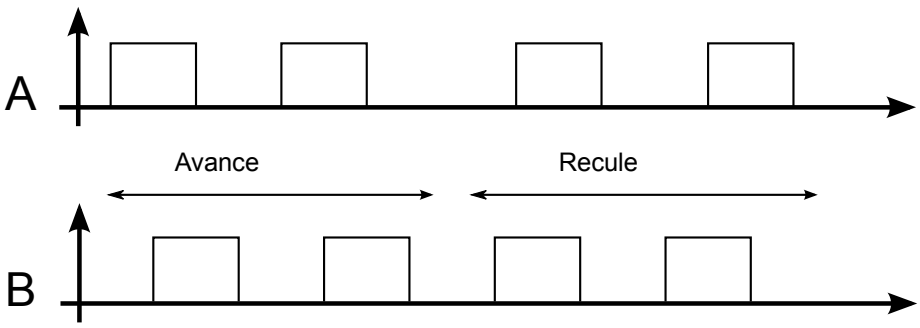
```
void loop () {
  if (Pous1On) { LedRougeOn) ; }
  if (Pous2On) { LedRougeOff) ; }
}
```

Récrivez un programme (*plus compliqué !)* qui fera la même chose, en appliquant la méthode proposée dans le cours :

- utilisez une **variable d'état**, qui sera initialisée à l'état initial (dans setup).
- dans la boucle principale (loop), sélectionnez **chaque état** possible (par une structure switch... case ou par des if...)
- pour chaque état, activez les **sorties** correspondantes
- détectez chaque **transition** dans l'état d'où elle part (par un test des conditions sur les entrées correspondantes) et modifiez alors la variable d'état.

2) Moteur pas-à-pas

La commande d'un **moteur pas-à-pas** simple peut se faire avec deux signaux déphasés :



Consultez http://fr.wikipedia.org/wiki/Moteur_pas_%C3%A0_pas pour avoir plus de détails

ou regardez le vidéo :http://pyr.ch/coursera/pas-a-pas.mp4

On voit que la séquence permet de choisir le sens de rotation.

a) Dessinez le graphe d'état du système. Remarquez que les transitions n'ont pas de condition : le moteur change de position chaque seconde (c'est l'horloge de la machine d'état).

Comme les deux sorties peuvent prendre 4 valeurs différentes (00, 01, 11, 10), la machine d'état aura 4 états.

b) Programmez la commande d'un moteur pas-à-pas qui avance ou recule d'un pas par seconde, selon une entrée ***Sens*** (dans un sens si on ne presse pas sur le poussoir, dans l'autre si on presse). Utilisez les LEDs rouges et vertes pour les sorties et le poussoir 1 pour le Sens.

Pour produire un changement de position par seconde, ajoutez une attente d'une seconde dans la boucle principale.

3) Gestion du temps

Tout en continuant d'appliquer la méthode d'écriture du programme d'une machine d'état, réalisez **une minuterie** pour l'éclairage d'un escalier. P*renez plutôt 3 secondes que 3 minutes pour vos tests...*

Dessinez le graphe d'état. Remarquez qu'une des transitions n'a pas une condition sur une entrée, mais sur la fin de la minuterie.

Votre programme peut se présenter de la manière suivante :

```
uint16_t timer = 0; // variable globale
void loop () {
  switch etat {
  case ...
  case ...
  }
  if (Timer >0) {
    Timer--;
  }
  delay (1);
}
```

La boucle principale est cadencée à 1 kHz (par le délai de 1ms). La variable Timer est décrémentée à chaque cycle si elle est supérieure à zéro.

Lorsqu'une transition doit correspondre au début d'une attente, la variable Timer est assignée à la durée de cette attente (*pour 3 secondes : Timer = 3000*). La transition qui correspond à la fin de l'attente s'écrit alors : if (Timer == 0) { etat = …} ;

4) Optionnel

Améliorez la commande de la perceuse, en faisant que le bouton ***Start*** devienne aussi un arrêt d'urgence durant la descente ou la montée de la perceuse. Une pression suivante sur ***Start*** fera alors remonter la perceuse.

Attention : de nouveaux états et éventuellement l'utilisation d'un timer vont être nécessaires. Par exemple, le fait que la perceuse soit dans l'état descente n'est pas suffisant pour que le bouton ***Start*** provoque l'arrêt : en effet le système passerait immédiatement à l'arrêt, avant même que le poussoir soit relâché !

La solution la plus propre est de ne déclencher le transitions du graphe d'état que lors des changements de l'entrée (en utilisant des variables du type valeur et ancienneValeur)