TP du 10 mars 2016

Graphes et machines d'état

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

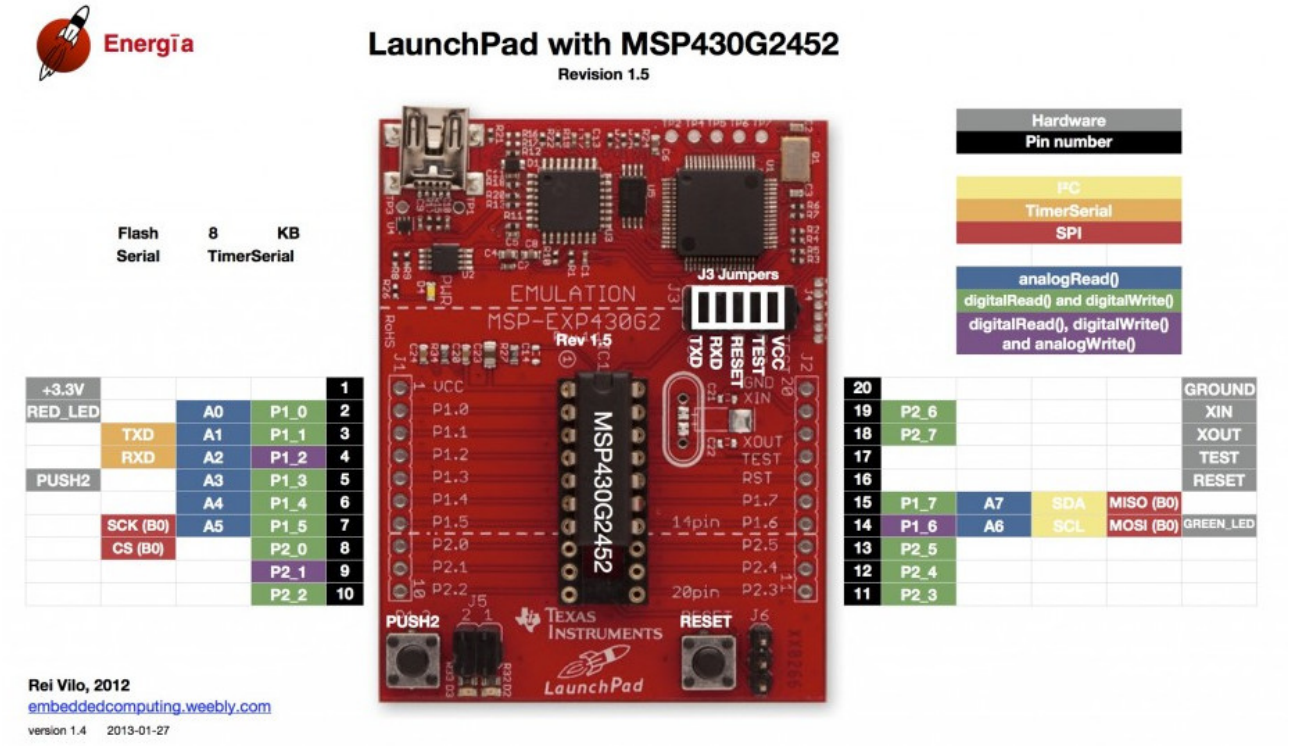
Pour disposer jusqu'à 3 boutons-poussoirs avec le LaunchPad, un circuit été préparé, qui se place sur le Launchpad. Les deux poussoirs supplémentaires sont sur P1.4 et P1.5. Ils peuvent être lus par des `digitalRead()`. Voici les définitions qui peuvent être utilisées :l

```
#define Pous2On (!digitalRead (P1_4))
#define Pous3On (!digitalRead (P1_5))
```

N'oubliez pas les initialisations correspondantes dans le `Setup()` :

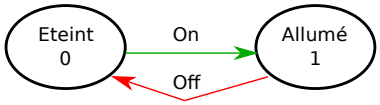
```
pinMode (P1_4, INPUT_PULLUP);
pinMode (P1_5, INPUT_PULLUP);
```

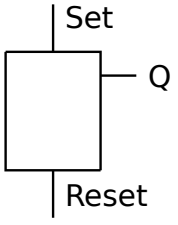
A titre d'information, voici un tableau très utile qui donne l'usage des pattes du LaunchPad :



1) <u>Bascule Set-Reset</u>

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 (Pous2On) { LedRougeOn) ; }
  if (Pous3On) { LedRougeOff) ; }
```

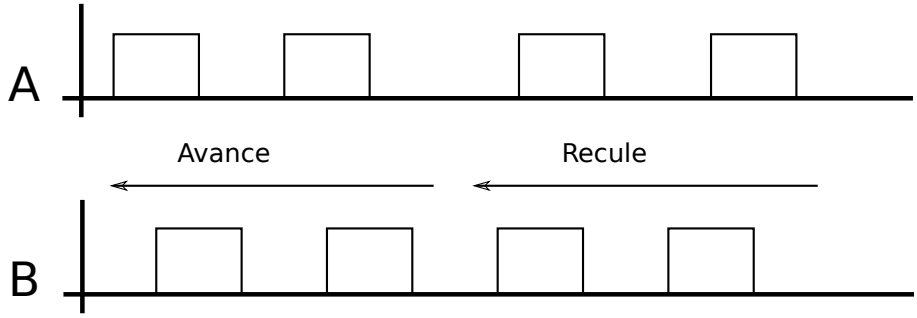
```
}
```

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

- utilisez une **variable d'état**
- dans la boucle principale, 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.

Dessinez le graphe d'état du système.

Programmez la commande d'un moteur pas-à-pas qui avance ou recule à vitesse fixe 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 P1.3 pour le Sens.

Pratique *: quelques exemplaires d'un montage avec un moteur pas-à-pas sera disponible durant le TP. Vous pourrez faire tourner le moteur ! Utilisez les pattes P2.0 et P2.1 pour connecter les fils Logidules.*

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 :

```
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) *(à rendre: envoyez votre code fonctionnel à pyr@pyr.ch)*

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 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 passerai immédiatement à l'arrêt, avant même que le poussoir soit relâché !