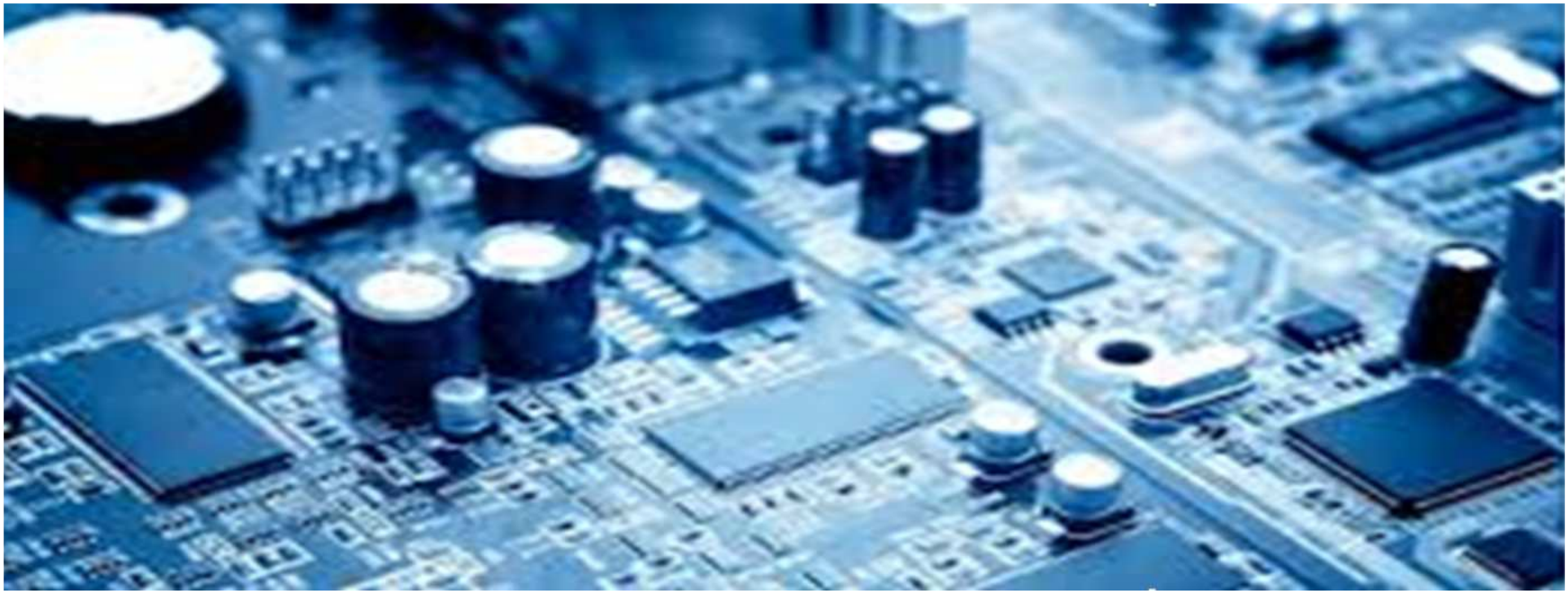


Gestion d’un menu par machine d’états

M1 Electronique

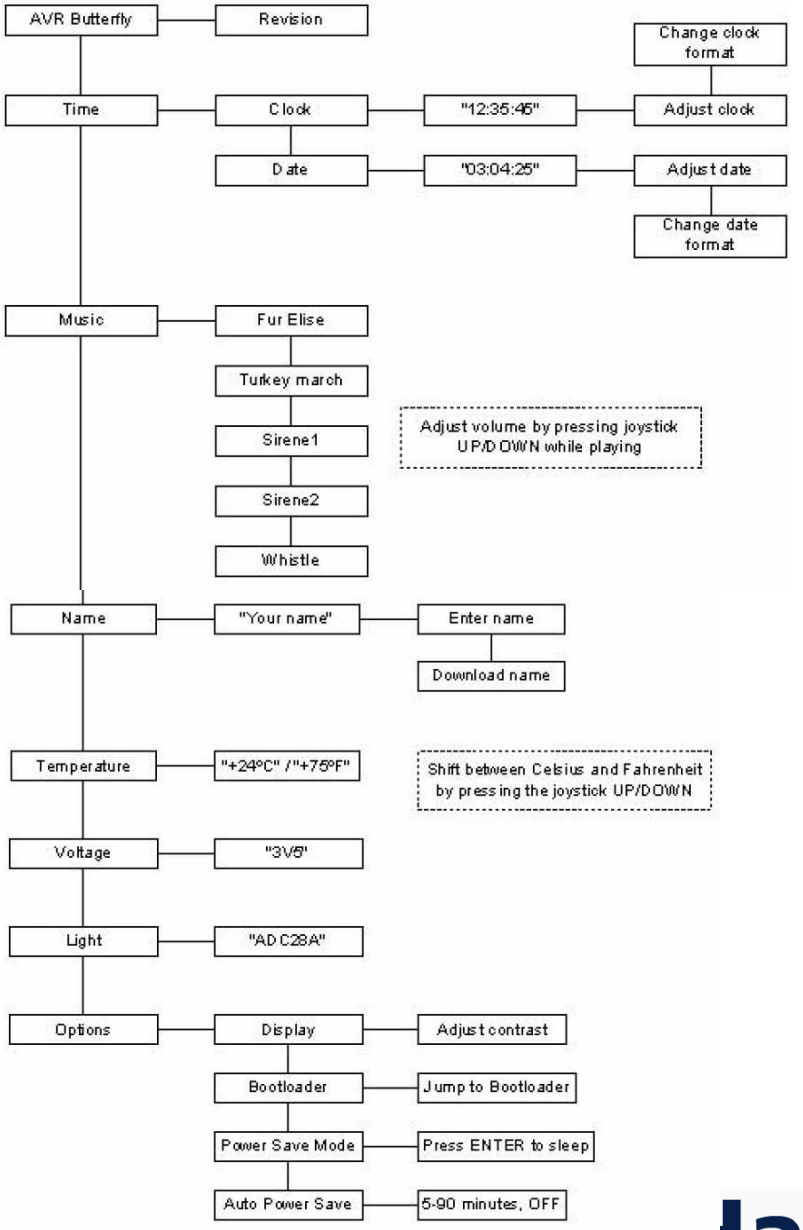
Haute Ecole Roi Louvain en Hainaut

Ref : Electronique Numérique

Date : 31/10/2013

HELHa | Haute École Louvain en Hainaut

Exemple prg Carte Butterfly



Fabrice Triquet

- Un état est constitué :
  - Soit de la visualisation d'un texte (interface utilisateur)
  - Soit d'un appel de fonction (via pointeur de fonction)

Un état ne peut pas être défini en même temps par un texte fixe et un pointeur de fonction. Il est toutefois possible d'afficher un texte à partir de la fonction.

Un état est défini par une structure composée :

- Du nom de l'état.
- Du texte qui lui est associé.
- Du pointeur de fonction qui lui est associé.

Un des champs texte ou pointeur de fonction devra être mis à null.

```
typedefstruct PROGMEM
{
unsigned char state;
PGM_P pText;
char (*pFunc)(char input);
}
MENU_STATE;
```

Fabrice Triquet

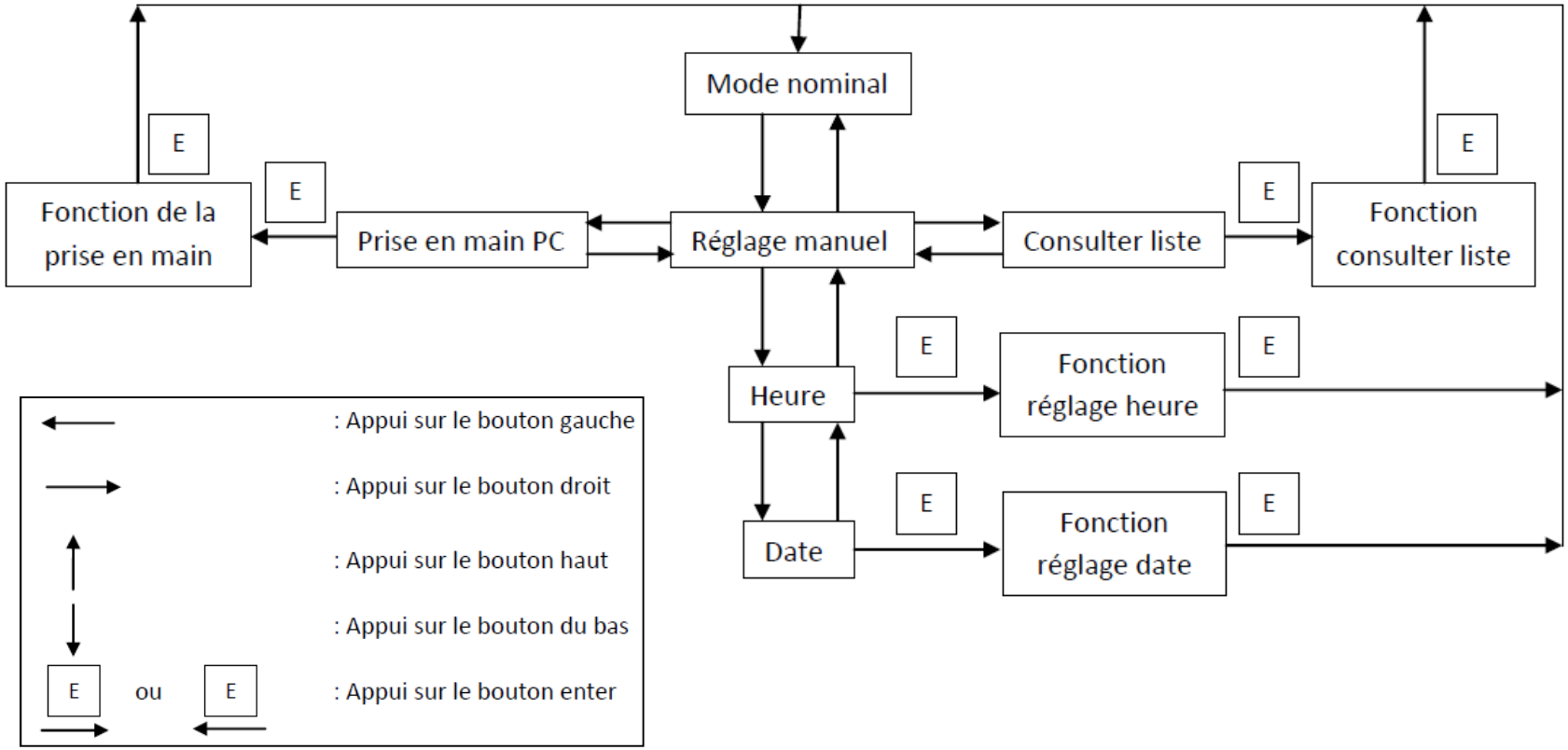
L'état suivant dépend de l'état dans lequel on est actuellement et de l'entrée qui est modifiée. On peut donc en tirer sa structure qui est la suivante :

- Le nom de l'état.
- L'entrée qui va permettre de modifier les états.
- Le nom de l'état suivant qui est associé à la modification de l'entrée.

```
typedefstruct PROGMEM
{
unsigned char state;
unsigned char input;
unsigned char nextstate;
}
MENU_NEXTSTATE;
```

Pour une meilleure utilisation du menu par l'utilisateur, il est préférable de définir un état nécessitant une validation avant de lancer la fonction. Ce premier état sera défini par un texte simple informant l'utilisateur sur la fonction qui sera appelée. On arrive, après validation, dans le second état contenant un pointeur de fonction.

Machine d'états
Exemple

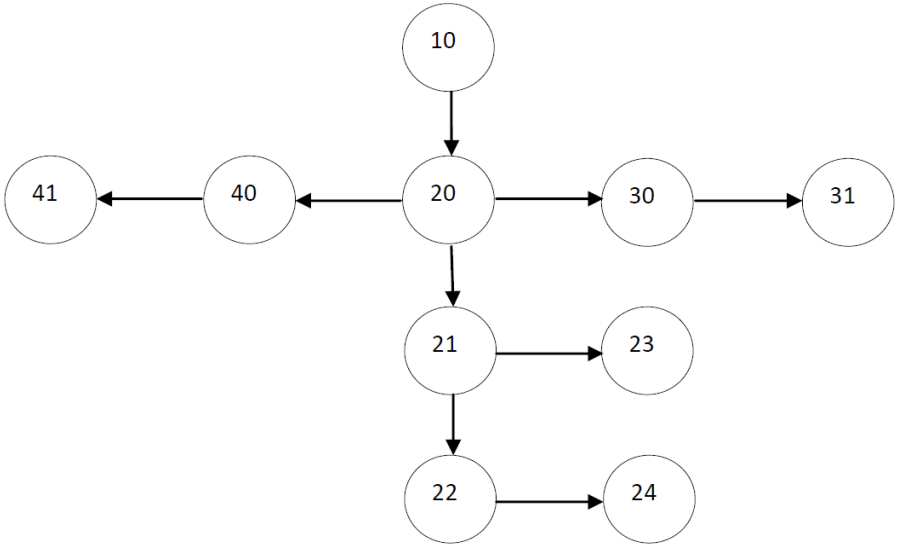


En appuyant sur les boutons gauche, droite, haut et bas, l'utilisateur peut se déplacer dans le menu parmi les différents états contenant un texte fixe. Une fois que l'option désirée a été choisie, il faut valider le choix par l'appui sur le bouton enter qui permet d'arriver dans un état contenant un pointeur de fonction.

Fabrice Triquet

Pour plus de facilité, il est conseillé d'associer chaque état possible à un nombre. Ceux-ci n'ont pas d'importance mais il est toutefois recommandé de mettre des chiffres de façon logique afin de visualiser directement l'état dans lequel on se trouve.

Il suffit de faire un « define » des différents états avec un nombre.



#defineMode_nominal	10
#defineReglage_manuel	20
#defineReglage_heure	21
#defineReglage_heure_func	23
#defineReglage_date	22
#defineReglage_date_func	24
#defineConsulter_liste	30
#defineConsulter_liste_func	31
#defineReglage_PC	40
#defineReglage_PC_func	41

Maintenant que nous avons l'ensemble des états, il faut programmer les différentes transitions qui sont possibles à partir de chacun des états. Pour cela, on utilise la structure nextstate, expliquée précédemment, en faisant correspondre les différents états suivants en fonction de l'état et de l'entrée associée.

Il est défini comme suit :

```
const MENU_NEXTSTATE menu_nextstate[] PROGMEM = {
{Nom de l'état,          Entrée,          Nom de l'état suivant qui est associé},
```

On définit l'ensemble des correspondances entre les états ainsi que le cas par défaut.

Il peut donc n'y avoir qu'un seul état suivant pour une seule entrée mais plusieurs états suivants pour différentes entrées. De plus, un état possédant un pointeur de fonction peut avoir comme état suivant n'importe quel état car la fonction comporte un return vers un état.

Machine d'états
Transitions

```
const MENU_NEXTSTATE menu_nextstate[] PROGMEM = {
//  STATE                INPUT                    NEXT STATE
// Mode nominal
{Mode_nominal,            KEY_DOWN,                Reglage_manuel},
// Réglage PC
{Reglage_PC,              KEY_RIGHT,               Reglage_manuel},
{Reglage_PC,              KEY_ENTER,               Reglage_PC_func},
// Réglage Manuel
{Reglage_manuel,          KEY_UP,                  Mode_nominal},
{Reglage_manuel,          KEY_RIGHT,               Consulter_liste},
{Reglage_manuel,          KEY_DOWN,                Reglage_heure},
// Consuter la liste
{Consulter_liste,         KEY_LEFT,                Reglage_manuel},
{Consulter_liste,         KEY_ENTER,               Consulter_liste_func},
// Reglageheure
{Reglage_heure,           KEY_UP,                  Reglage_manuel},
{Reglage_heure,           KEY_ENTER,               Reglage_heure_func},
{Reglage_heure,           KEY_DOWN,                Reglage_date},
// Régler la date
{Reglage_date,            KEY_UP,                  Reglage_heure},
{Reglage_date,            KEY_ENTER,               Reglage_date_func},
// Par défaut
{0,                       0,                       0}
};
```

Fabrice Triquet

Il est aussi nécessaire de faire une table permettant de savoir quel texte ou pointeur de fonction est associé à chacun des états. Comme expliqué précédemment, un état ne peut être composé que d'un texte ou d'un pointeur de fonction. Une de ces deux valeurs doit être mise à null.

```
const MENU_STATE menu_state[] PROGMEM = {
//  STATESTATE TEXT                         STATE_FUNC
{Etat,                    texte fixe de l'état,              pointeur de fonction de l'état},
```

```
const MENU_STATE menu_state[] PROGMEM = {
//  STATE          STATE TEXT                                STATE_FUNC
{Mode_nominal,                Texte_nominal,                          NULL},
{Reglage_PC,                  Texte_reglage_pc,                       NULL},
{Reglage_PC_func,             NULL,                                   fonction_pc},
{Reglage_manuel,              Texte_reglage_manuel,                   NULL},
{Consulter_liste,             Texte_consulter_liste,                   NULL},
{Consulter_liste_func,        NULL,                                   Lire_liste},
{Reglage_heure,               Texte_reglage_heure,                    NULL},
{Reglage_heure_func,          NULL,                                   Set_clock},
{Reglage_date,                Texte_reglage_date,                      NULL},
{Reglage_date_func,           NULL,                                   Set_date},
// Defaut
{0,                            NULL,                                  NULL},
};
```

Les textes fixes doivent être introduits dans la mémoire du programme sous forme de string.

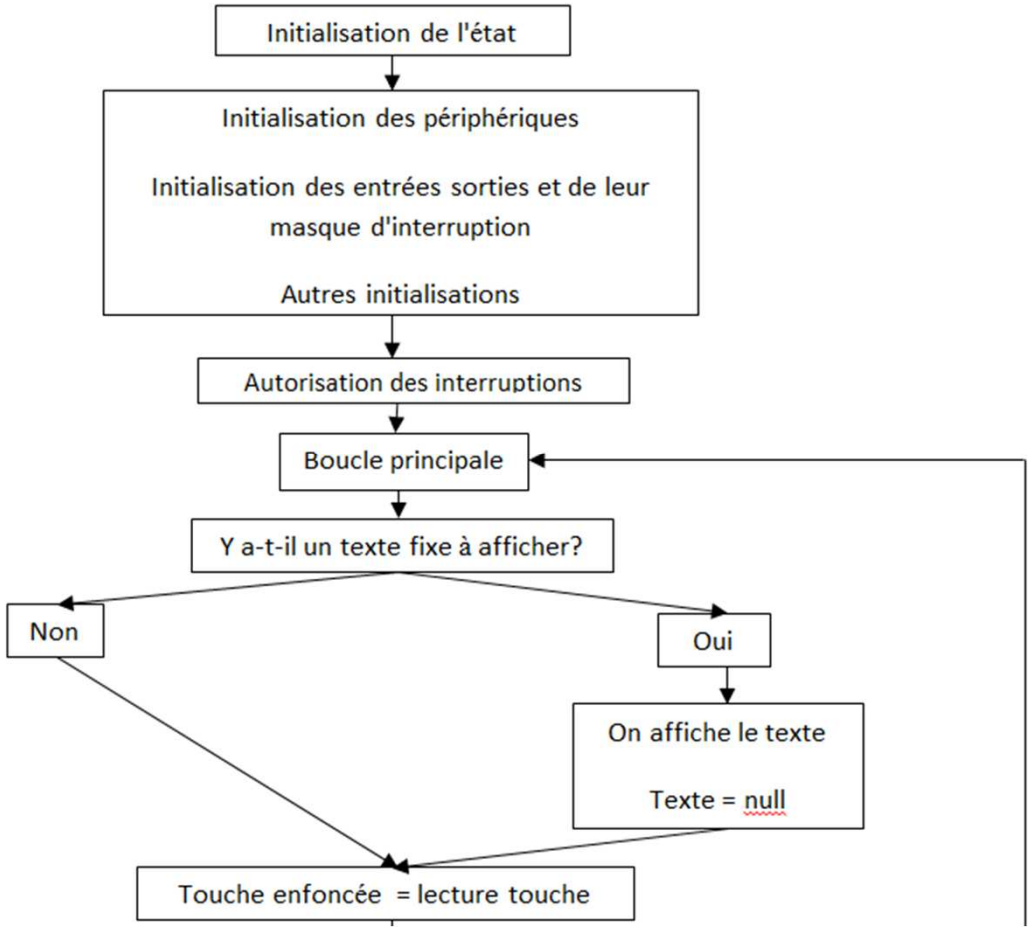
Ils sont définis sous forme de :

```
const char nom_du_texte_defini_pour_l'état[] PROGMEN = "Le texte que l'on désire";
```

```
const char Texte_reglage_heure[] PROGMEM          = "HEURE";
const char Texte_reglage_date[] PROGMEM           = "DATE";
const char Texte_reglage_manuel[] PROGMEM         = "REGLAGE MANUEL";
const char Texte_consulter_liste[] PROGMEM        = "CONSULTER LA LISTE";
const char Texte_reglage_pc[] PROGMEM             = "REGLAGE PC";
const char Texte_nominal[] PROGMEM                = "MODE NOMINAL";
```

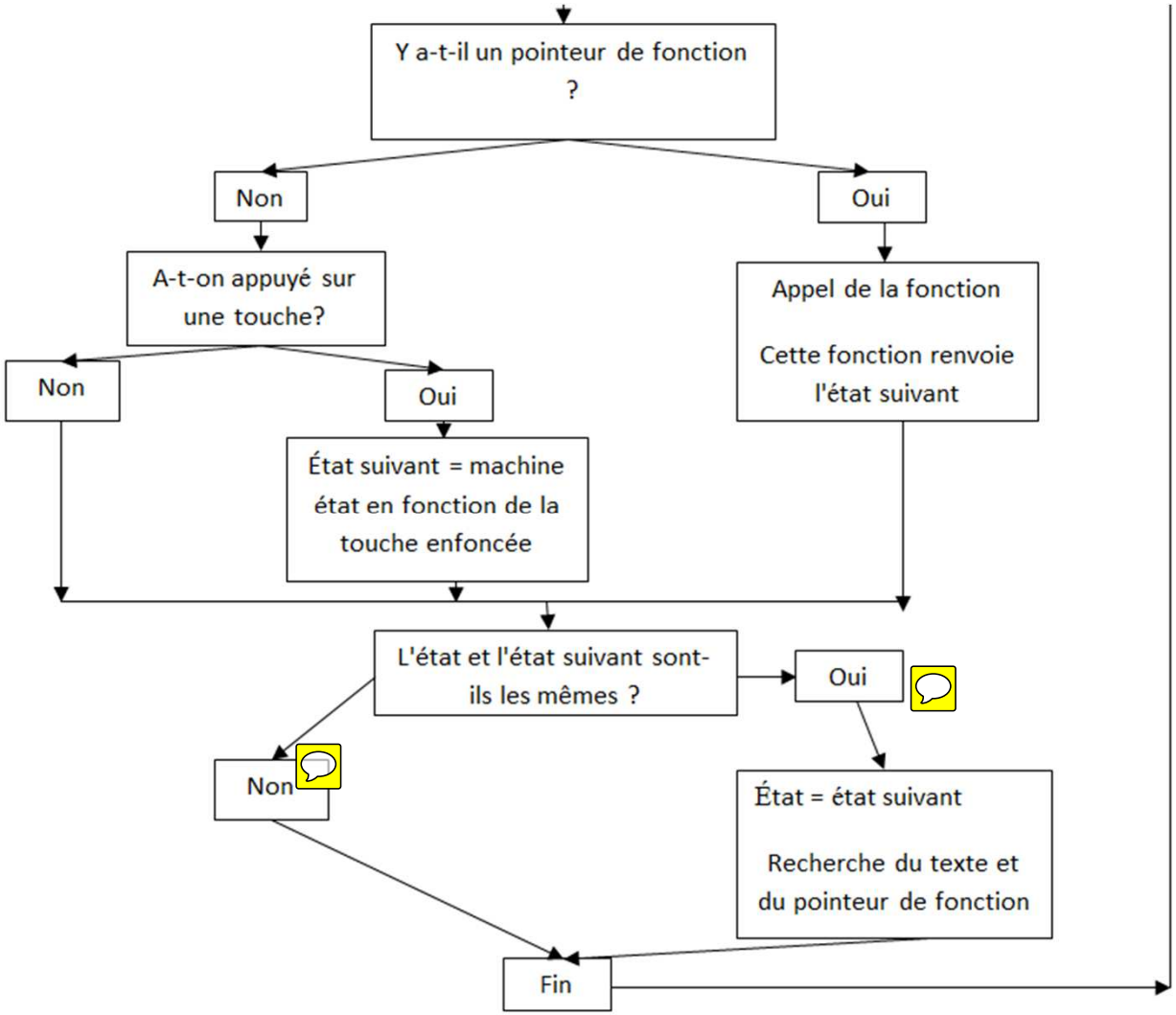
Fabrice Triquet

Machine d'états



Fabrice Triquet

Machine d'états



Fabrice Triquet

Bibliographie

Fabrice Triquet