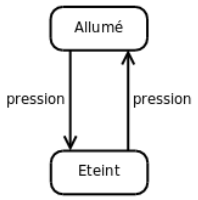
Développement logiciel

Implémentation d'une machine à états

Application IHM minuteur électronique

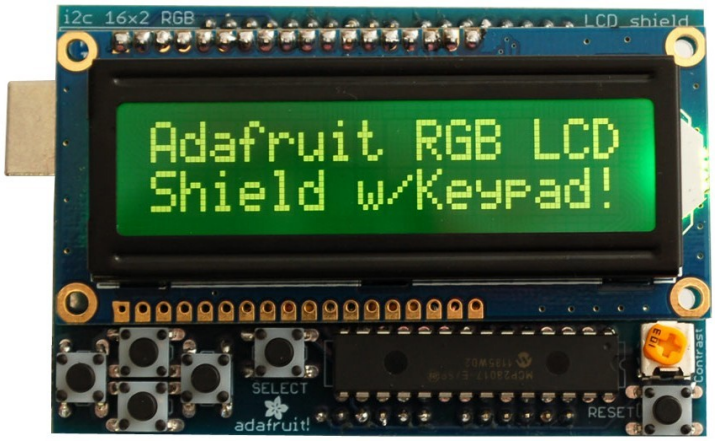


Objectif : Être capable de développer une application simple sur un module Arduino en utilisant le formalisme de la machine à états. L'étude s'appuie sur une application d'IHM de minuteur électronique utilisant un écran lcd et un clavier basique (4 boutons + select).

1. Présentation de l'application à coder

Vous allez réaliser la structure du programme gérant l'affichage d'un minuteur électronique en utilisant le formalisme de la machine à états.

La maquette utilisée comprend une carte Arduino Uno équipée d'un shield Adafruit RGB LCD (carte d'extension). C'est un afficheur LCD comprenant 16 caractères sur 2 lignes et un clavier de 5 boutons (← ; ↑ ; → ; ↓ ; SEL).

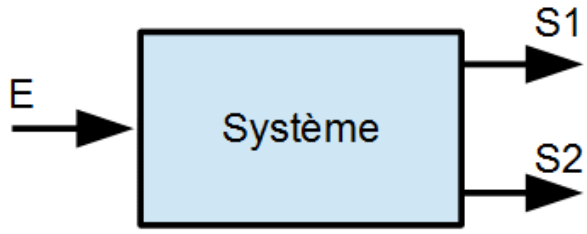


Le cahier des charges est le suivant : réaliser l'interface homme-machine (IHM) d'un minuteur électronique programmable de 0 à 59 min et 59 secondes par pas de 1 s.

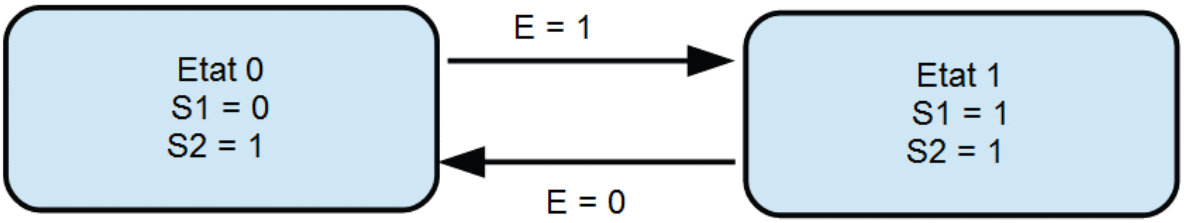
Au cours de ce TP, on ne s'intéressera qu'à l'affichage des différents menus.

2. Rappel : codage d'une machine à états en C

On rappelle ici la représentation d'une machine à états. Voici un système avec 2 sorties et une entrée :



La machine à états qui représente son fonctionnement est la suivante :



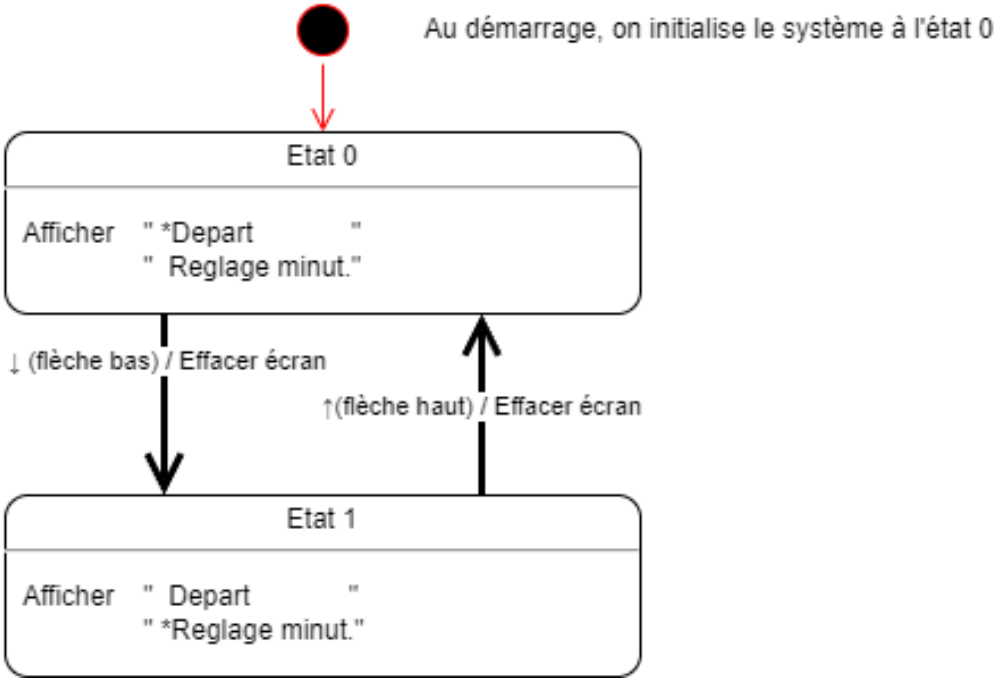
Celle-ci est sera codée comme suit en C :

```
int Etat = 0 , S1 = 0, S2 = 0 ; E = 0 ;
while (1)
{// debut while
    switch(Etat)
    {
    case 0 :
        { // debut case 0
            S1 = 0 ;
            S2 = 1 ;
            if (E==1)
            { Etat = 1 ; }
         } // fin case 0
    break ;
    case 1 :
        { // debut case 1
            S1 = 1 ;
            S2 = 1 ;
            if (E==0)
            { Etat = 0 ; }
        } // fin case 1
    break ;
    } // fin switch - case
} // fin while
```

3. Codage d'une première machine

Dans un premier temps, nous allons implémenter en C dans l'Arduino Uno, une machine à états permettant de gérer l'affichage des deux premiers menus du minuteur : le choix entre le départ du minuteur et le réglage du minuteur.

Voici représenté les deux états du système avec les écrans que l'on souhaite afficher.



On remarquera sur ce diagramme :

- On a défini ici l'état initial de la machine à états lorsque le système démarre. La variable d'état doit être initialisée à 0 (etat = 0 au démarrage).
- La présence d'action associée à des transitions. Lorsque l'utilisateur appuie sur ↓ (flèche bas), on efface l'écran.

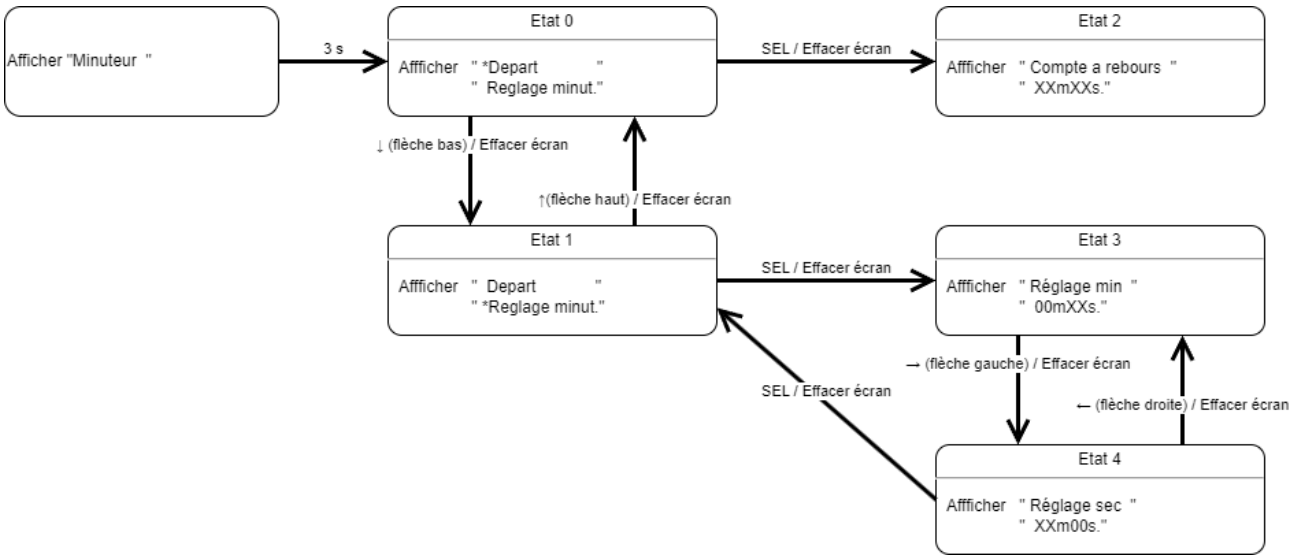
Expérimentation

Q.1) Compléter le fichier donné en annexe (fichier fourni) afin d'y faire figurer l'état 1.

Q.2) Valider le fonctionnement.

4. Implémentation des menus du minuteur

Vous allez maintenant implémenter tous les menus d'affichage du minuteur conformément à la figure suivante.



Travail demandé

Q.1) Implémenter le programme permettant de gérer l'affichage conformément au diagramme donné. Remarque : compte tenu de la structure de la machine à états, vous pouvez commencer par coder l'état 2 , tester puis implémenter après les états 3 et 4.

Q.2) En bonus, si vous êtes en avance, mettre en œuvre le réglage des minutes et des secondes (état 3 et 4) puis le compte à rebours dans l'état 2.

Remarque : Le premier écran (Minuteur) sera affiché dans le « setup » et n'est affiché que durant 3 secondes.

5. Annexe 1 : programme de base à compléter

```
#include <Adafruit_RGBLCDShield.h>

Adafruit_RGBLCDShield lcd = Adafruit_RGBLCDShield();

// These #defines make it easy to set the backlight color
#define WHITE 0x7

int etat = 0 ;
uint8_t buttons = 0 ;

void setup() {
  // Debugging output
  Serial.begin(9600);
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  lcd.setBacklight(WHITE);
}


void loop() {
  switch(etat)
  { // debut switch
  case (0) :
  {
    lcd.setCursor(0, 0);
    lcd.print(" *Depart ");
    lcd.setCursor(0, 1);
    lcd.print("  Reglage minut.");
    buttons = lcd.readButtons();
     if (buttons & BUTTON_DOWN)
       {  etat = 1 ;
       lcd.clear() ;}
  } // fin case 0
  break ;

  //
  // A completer
  //

} // fin switch - case
} // fin loop
```