Automate ARDUINO : 1 BP, 2 LEDs appui court – appui long

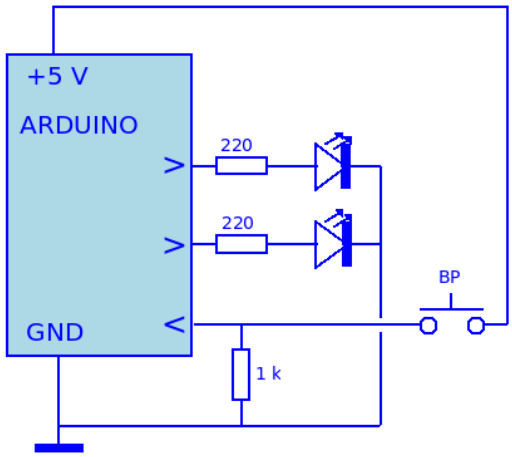
Description matérielle

L'automate est réalisé avec une carte ARDUINO UNO.

Périphériques d'entrée :
- un bouton poussoir NO (Normalement Ouvert) connecté à l'ARDUINO de façon à donner une entrée LOW quand il est ouvert, et une entrée HIGH quand il est fermé.

Périphériques de sortie :
- deux LEDs, une rouge et une verte.

Montage électrique



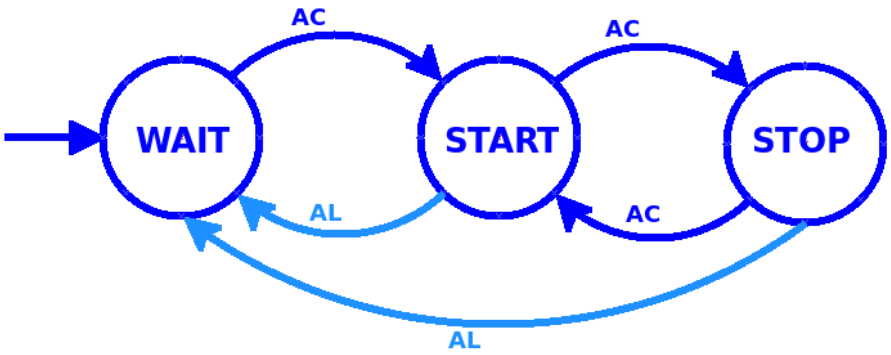
Fonctionnement

Le fonctionnement attendu est le suivant :

Le fonctionnement attendu est le suivant:

- à l'état initial, l'automate est à l'état **WAIT** : la LED rouge est éteinte et la LED verte est allumée.
- un appui court (événement **AC**) sur le bouton poussoir fait passer l'automate dans l'état **START** : la LED rouge s'éteint, la LED verte se met à clignoter,
- un nouvel appui court sur le bouton poussoir fait passer l'automate de l'état **START** à l'état **STOP** : la LED rouge s'allume, la LED verte s'éteint,
- les appuis courts successifs font passer successivement de l'état **START** et l'état **STOP**, et de l'état **STOP** et l'état **START**.
- Un « appui long» de plus de 3 seconde (événement **AL**) fait passer l'auomate à l'état **WAIT** quel que soit son état courant (**STOP** ou **START**).

Diagramme des états



AC : appui court bouton poussoir
AL : appui long bouton pousssoir.

Tableau des transitions d'états

C'est le tableau qui liste toutes les transitions d'états possibles décrites par le diagramme des états :

État de départ	*Événement*	*État d'arrivée*	*Action de transition à faire (une seule fois à la transition d'états)*
WAIT	AC	START	
START	AC	STOP	Allumer LEDR & éteindre LEDV
STOP	AC	START	Éteindre LEDR
START	AL	WAIT	allumer LEDV
STOP	AL	WAIT	Éteindre LEDR & allumer LEDV

Cette formulation du diagramme d'état sous la forme de tableau est directement utilisée pour la programmation arduino.

Tableau des états de l'automate

C'est le tableau qui montre **ce que l'automate doit faire de façon permanente** avec ses différents périphériques de sortie (LEDs, moteurs, actionneurs, alarmes...) dans ses différents états :

État automate	**LED rouge**	**LED verte**
WAIT	RAF	RAF
START	RAF	Faire clignoter
STOP	RAF	RAF

Algorithme textuel

```
//
// Déclaration des variables globales :
//
variables globales :
        etat      (l'état de l'automate)
        oldEtatBP (état mémorisé du Bouton Poussoir 'BP')
        t0        (instant appui BP)
constantes :
        WAIT    <- 1    (constante désignant l'état 'wait')
        START   <- 2    (constante désignant l'état 'start')
        STOP    <- 3    (constante désignant l'état 'stop')
        pinLEDR <- 5  (la broche de la LED Rouge )
        pinLEDV <- 6  (la broche de la LED Verte )
        pinBP     <- 7  (la broche du bouton poussoir)
configurer :
    pinBP en entrée,
    pinLEDR et pinLEDV en sortie

// Initialisation des variables globales qui le nécessitent :
oldEtatBP <- LOW

// Définition de l'état initial :
état <- WAIT
éteindre LEDR
allumer  LEDV

loop:    // pour ARDUINO, la boucle est assurée par la fonction loop

    // 1 - lecture des périphériques d'entrée qui créent les événements de l'automate.…
    //     (→ exploitation du diagramme des états)

    etatBP <- lecture de l'état du Bouton Poussoir

    // 2 - Traitement des événements provoquant une transition d'état :
    //     (→ exploitation du Tableau des transitions d'état)

    //   Événement AC : Appui Court sur Bouton Poussoir (BP)
    //   Événement AL : Appui Long ( ≥ 3 sec) sur BP

   si oldEtatBP est LOW ET etatBP est HIGH :              // Appui sur BP => Front montant
       | t0 <- temps initial
   sinon si oldEtatBP est HIGH ET etatBP est LOW :          // BP relâché => Front descendant
       | t1 <- temps courant
       | si t1 - t0 < 3 secondes:  // événement AC (Appui Court) détecté
       |    |si etat est WAIT :
       |    |    | etat <- START
       |    |sinon si etat est START :
       |    |    | etat <- STOP
       |    |    | allumer la LEDR et éteindre la LEDV
       |    |sinon si etat est STOP :
       |    |    | etat <- START
       |    |    | éteindre la LEDR
   sinon si oldEtatBP est HIGH ET etatBP est HIGH :          // BP appui maintenu …
       | t1 <- temps courant
       | si t1 - t0 >= 3 secondes:  // événement AL (appui long) détecté
       |    | si état est STOP
       |    |    | état <- WAIT
       |    |    | éteindre LEDR et allumer LEDV
       |    | si état est START
       |    |    | état <- WAIT
       |    |    | allumer LEDV

    // 3 - Traiter les actions (permanentes) à faire pour les états possibles :
    //     (→ Exploitation du Tableau des états)
    si  etat est  WAIT :
        | RAF (Rien A Faire)
    sinon si etat est  START:
        | faire clignoter la LED V : mettre la LEDV dans l'état opposé à son état actuel
        | (allumée -> éteinte ou éteinte-> allumée)
    sinon etat est STOP:
        | RAF

   mémoriser etatBP dans oldEtatBP
   attendre "un peu"

fin loop
```

Programme ArduinO

unBPdeuxLED-AL

```
1  ////////////////////////////////////////
2  // Les macros et variables globales     //
3  ////////////////////////////////////////
4
5  #define pinLEDR 5
6  #define pinLEDV 6
7  #define pinBP   7
8
9  #define WAIT  1
10 #define START 2
11 #define STOP  3
12
13 int state;            // L'ETAT de l'automate
14 int old_state_BP;     // Mémorisation de l'état du Bouton Poussoir (BP)
15 long unsigned int t0; // instant appui BP
16
17 void setup()
18 {
19   // Configutaion des E/S:
20   pinMode(pinBP, INPUT);
21   pinMode(pinLEDR, OUTPUT);
22   pinMode(pinLEDV, OUTPUT);
23
24   // au démarrage, mettre l'automate dans l'état WAIT
25   state = WAIT;
26   digitalWrite(pinLEDR, LOW);
27   digitalWrite(pinLEDV, HIGH);
28
29   // initialidsation variables:
30   old_state_BP = LOW;
31 }
32

33 void loop()
34 {
35   // 1 - lecture des périphériques d'entrée qui peuvent créer
36   //     les événements de l'automate....
37   //     (exploitation du diagramme des états)
38
39   int current_state_BP = digitalRead(pinBP);
40
41   // 2 - Traitement des événements provoquant une transition d'état :
42   //     (-> exploitation du Tableau des transitions d'état)
43
44   if (old_state_BP == LOW && current_state_BP == HIGH) // Front montant
45   {
46     t0 = millis();
47   }
48   else if (old_state_BP == HIGH && current_state_BP == LOW) // Front descendant :BPA
49   {
50     long unsigned int t1 = millis();
51     if (t1 - t0 < 3000)
52     {
53       // Événement BPA détecté:
54       if (state == WAIT)
55       {
56         state = START;
57       }
58       else if (state == START)
59       {
60         state = STOP;
61         digitalWrite(pinLEDR, HIGH);
62         digitalWrite(pinLEDV, LOW);
63       }
64       else if (state == STOP)
65       {
66         state = START;
67         digitalWrite(pinLEDR, LOW);
68       }
69     }
70   }
```

```
  else if (old_state_BP == HIGH && current_state_BP == HIGH) // Appui maintenu : AL
  {
    long unsigned int t1 = millis();
    if (t1 - t0 >= 3000)
    {
      // événement AL (Appui Long)
      if (state == STOP)
      {
        state = WAIT;
        digitalWrite(pinLEDR, LOW);
        digitalWrite(pinLEDV, HIGH);
      }
      else if (state == START)
      {
        state = WAIT;
        digitalWrite(pinLEDV, HIGH);
      }
    }
  }


  //
  // 3 - Traitement des états
  //

  if (state == WAIT)
  {
     ;  // RAF
  }
  else if (state == START)
  {
    // faire clignoter la LEDV
    int current_state_LEDV = digitalRead(pinLEDV);  // récupèrer l'état courant de la LEDV
    digitalWrite(pinLEDV, 1 - current_state_LEDV);  // lui donner l'état contraire
  }
  else if (state == STOP)
  {
      ; // RAF
  }


  //
  // autre staff à faire dans la boucle...
  //

  old_state_BP = current_state_BP;
  delay(100);
}
```