```
/************************************/
/* Programme de la carte de sécurité  */
/* INRIA Rhone-Alpes - service SED    */
/* COLOMBAN Romain Stage DUT 2004     */
/************************************/
 // définition du PIC
 #pragma chip PIC16F877
 #include "int16cxx.h"
 // définition des entrées-sorties
 #pragma char PORT_S @ PORTD
 #pragma char PORT_E @ PORTC
 #pragma char PORT_AU @ PORTB

 #pragma bit AR_UR @ PORT_S.4
 #pragma bit voy_marche @PORT_S.5
 #pragma bit rearm @ PORT_E.0

 char codage (void);
 char flags_AU,n,PORT_AU_Temp;

  ////////////////////////////////////
 /* programme d'interrpution          */
 ////////////////////////////////////
 #pragma origin 4
 interrupt int_server(void)
 {
    int_save_registers
    if(INTE){                  //front sur B0 (Arrêt d'urgence global)
    if(INTF){
                INTE = 0;
                INTF = 0;
            PORT_S =codage();  ////// Mise à jour sorties //////
                AR_UR = 1;
        }}
        int_restore_registers
 }
 // placer les include ici
 #include "delay.h"

 ////////////////////////////////////
 /* codage de l'erreur                */
 ////////////////////////////////////
 char codage(void)
 {
    flags_AU = 0;
    PORT_AU_Temp = PORT_AU & 0xFE;
        n = 0;         ////// Test entrée active //////
                if(PORT_AU_Temp !=0)
                {
                        while(!PORT_AU_Temp.0)
                        {
                                n++;
                                PORT_AU_Temp /= 2; //décalage à droite
                        }
                }
                switch (n)    ////// Codage sorties //////
                {
                case 1:
                        flags_AU = 15;      //normalement rien
                        break;
                case 2:
                        flags_AU = 1;      //Arrêt logiciel
                        break;
                case 3:
                        flags_AU = 2;      //Drive Enabled
                        break;
                case 4:
                        flags_AU = 4;      //Arrêt d'urgence
                        break;
                case 5:
                        flags_AU = 9;      //FDC_Pelvis
                        break;
                case 6:
                        flags_AU = 10;     //FDC_jDroite
                        break;
                case 7:
                        flags_AU = 12;     //FDC_jGauche
```

```
78                          break;
79                  default:
80                          flags_AU = 15;      //mauvaise détection
81                          break;
82                  }
83          return flags_AU;
84    }
85
86    ////////////////////////////////////
87    /* programme principal              */
88    ////////////////////////////////////
89    void main(void)
90   {
91    OPTION = bin(11000000);
92    ADCON1 = 0x06;                    //pas d'analogique
93
94    PORTA = 0;
95    TRISA = 0;
96    PORTB = 0;
97    TRISB = 0xFF;
98    PORTC = 0;
99    TRISC = 0x01;
100   PORTD = 0;
101   TRISD = 0;
102
103   AR_UR = 0;
104   flags_AU = 0;
105
106   DelayMs(255);     //tempo de 2s au démarrage
107   DelayMs(255);
108   DelayMs(255);
109   DelayMs(255);
110   DelayMs(255);
111   DelayMs(255);
112   DelayMs(255);
113   DelayMs(255);
114
115   INTF = 0;
116   INTE = 1;                         //interruption RB0 active
117   GIE = 1;                  // Global interrupt enable
118
119   if(PORT_AU !=0)                   //test au démarrage
120   {
121      INTF = 1;
122   }
123   while(1)
124   {
125      if(!flags_AU)             ////// Arrêt d'urgence ? //////
126      {                                 //oui
127          if(rearm)             ////// Réarmement ? //////
128          {                             //oui
129      /*while(rearm){}              //attente rearm = 0   */
130              if(!PORT_AU)      ////// Plus de défauds ? //////
131              {                         //oui
132          flags_AU = 0; ////// Redémarrage du robot //////
133          PORT_S = 0;
134          INTF = 0;
135                  INTE = 1;
136          }
137              else                      //non
138              {
139                  PORT_S = codage();
140                  AR_UR = 1;
141              }
142          }                             //non
143      voy_marche=!voy_marche;  ////// Clignottement voyant //////
144          DelayMs(255);
145       }
146      else                              //non(normal, mettre ici le test des potentiomètres)
147      {
148          voy_marche = 1;          ////// Allume voyant //////
149      }
150   }
151  }
```