Le Générateur PicCleaner du crocodile

Version : V1.0
Date : 10 juillet 2016
Auteur : croco31

Résumé :
Ce document décrit la construction d'un générateur de tension type « PolarCleaner » autonome à base de PIC12F675 dont le nombre de cycles est réglable de 1 à 10 cycles, chaque cycle durant 8x11minutes.

 **Avertissement** : 
Les informations données ici sont destinées à la réalisation expérimentale d'un montage électronique. L'auteur ne suppose aucune application thérapeutique de ce générateur et décline toute responsabilité suite à son usage.

1 Introduction

Le montage est basé sur les travaux du guérisseur américain Lee CROCK, travaux présentés en 2004 par Michel DOGNA sur son site (https://www.micheldogna.fr/polar-cleaner-serie-article-7-25-121.html), qui vante les mérites de cet appareil pour la dépollution du corps.
Cet appareil est disponible commercialement via la société ARKTIKAIA ou VALEMIS :
(http://www.arktikaia.com/technologie/technologie/polar-cleaner-s3-machine-a-laver-les-cellules-8.html).
http://www.valemis.com/

Intrigué par cet appareil et voulant tester ses effets, je fus un peu rebuté par le coût non négligeable de celui-ci pour un simple essai (plus de 500^{E}) et j'ai préféré construire un petit montage appliquant le principe de Lee CROCK.

Ce montage utilise un PIC programmé avec le logiciel adéquat afin de simplifier au maximum sa construction (une fois le PIC programmé bien sûr) et réduire son coût à quelques euros.

D'autres montages ont été réalisés et sont disponibles sur le web, souvent basés sur un programmateur et/ou relais, demandant une alimentation secteur supplémentaire.

Le montage montré ici utilise directement le bloc de 5x2 piles salines 1.5V pour alimenter le montage en +3V, le rendant de ce fait autonome.

2 Le principe de base

Les vertus alléguées de l'appareil sont basées sur une variation périodique du potentiel électrique d'une grille (cuivre ou alu) placée sous le drap du lit, ce potentiel étant le 0V ou le 3V d'un groupe de 5x2 piles SALINES (il est indiqué que c'est important bien que le PolarCleaner du commerce n'en utilise pas du tout mais un accu à la place semble-t-il).

Le potentiel appliqué à la grille alterne toutes les 11 minutes, ce qui correspond à une « respiration cellulaire », avec 1 ou plusieurs cycles de 8x11 minutes, le cycle se terminant sur la polarité +3V.

Il faut noter qu'un seul fil relie l'appareil à la grille, de façon similaire au négativeur LAVILLE, ce qui fait que l'appareil ne débite aucun courant, à part peut-être les courants de fuite (pA ou nA).

Afin de rendre l'appareil autonome et éviter le recours à une alimentation supplémentaire pour le circuit de commande, celui-ci est alimenté directement par le bloc de piles, en profitant du fait que le PIC utilisé fonctionne de 2 à 5V sans problème.
Il faut donc que l'électronique consomme très peu pour ne pas vider trop vite les piles (grosses piles type D).

3 L'électronique

Le schéma joint en annexe comprend :

- Un PIC12F675 qui assure toute la gestion des temporisations et l'interface avec l'utilisateur via le potentiomètre et la LED (rouge obligatoire à cause de sa faible tension de fonctionnement de 1.6V, ce qui n'est pas le cas des autres couleurs). Les sorties de commande des bobines du relais sont protégées par des diodes vers le +3V de manière classique. L'oscillateur interne du PIC calibré à 1% est suffisant pour régler les durées des cycles de 11 minutes.
- Un potentiomètre de réglage permettant de choisir le nombre cycles de 8x11 minutes à effectuer
- La LED de signalisation
- Un relais bistable AXICOM (code RadioSpares :718-1780) commandé par impulsions sur deux bobines séparées et fonctionnant en 3V ou moins. Le fait d'utiliser un relais bistable permet de le commander par impulsion de 3ms ce qui réduit énormément la consommation, et l'usage de deux bobines séparées permet de le commander par 2 broches directes du PIC. Ce relais applique la borne +3V ou la borne 0V du bloc de piles sur la grille placée sous le drap. Dans le cas d'usage du relais, la broche 4 (GP3) du PIC doit être connectée au 0V.
- Il est possible de se passer du relais (et de ses diodes schottky 1N5818 ou 1N4148 associées) en connectant la broche 4 du PIC au +3V et en connectant directement la grille sur la broche 5 du PIC. Dans ce cas c'est le PIC lui-même qui applique le +3V ou le 0V via son port de sortie ce qui revient au même. Dans ce cas la broche 2 peut être utilisée pour commander la base d'un transistor inverseur pilotant un relais normal (ce qui peut être utile si on alimente le montage par une autre source autre que le bloc de piles).
- L'interrupteur général de préférence bipolaire qui isole complètement le bloc de piles quand l'appareil est mis hors tension.

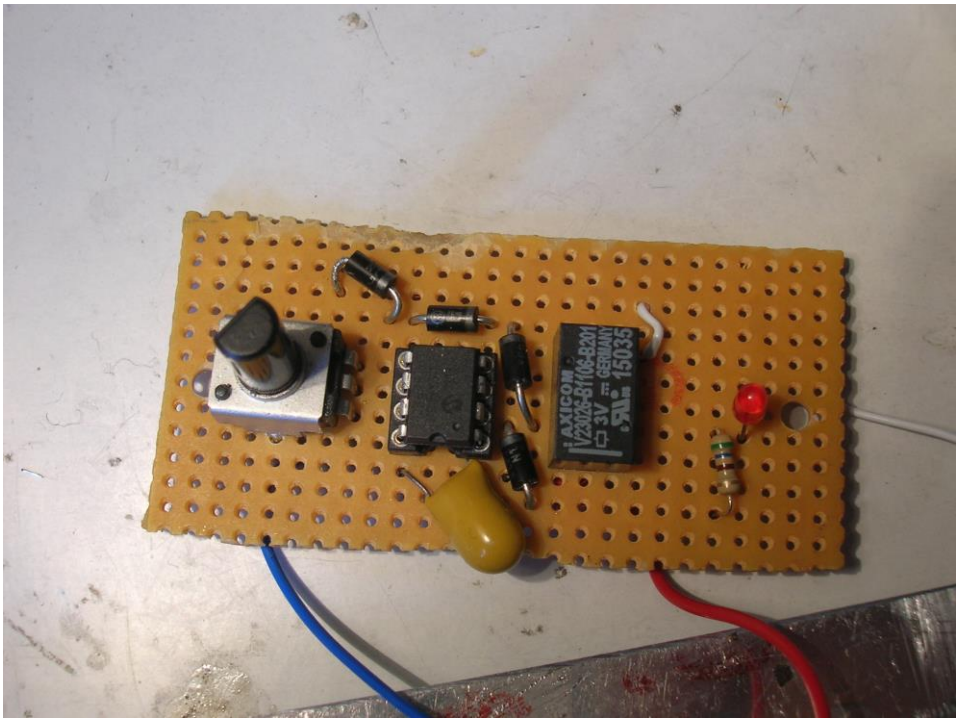
4 Le logiciel du PIC

Le logiciel est écrit en C sur MPLAB-XC8 (version gratuite disponible sur le site www.microchip.com) et comprend un fichier .c et un fichier .h joints en annexe.

Le fichier .HEX de programmation obtenu est joint en annexe sous format texte qu'il suffit de copier/coller dans un fichier texte brut pour l'utiliser.

5 Construction

Le montage a été assemblé sur une simple plaque proto à trous et un boitier au bon format en bois a été réalisé pour contenir le bloc de 2x5 piles au format D.



Le circuit

Le boitier

6 Utilisation

L'appareil est simplement démarré par l'interrupteur double, ce qui fait clignoter plusieurs fois la LED rouge pour indiquer que le logiciel a démarré, puis celui-ci affiche le nombre de cycles de 8 x11minutes qui seront effectués, comme programmé par le potentiomètre (10 zones de 1 à 10).

En cours de fonctionnement un flash très bref est affiché sur la LED rouge à chaque seconde, et le nombre de cycles programmé est réaffiché à chaque fin de période 11 minutes.

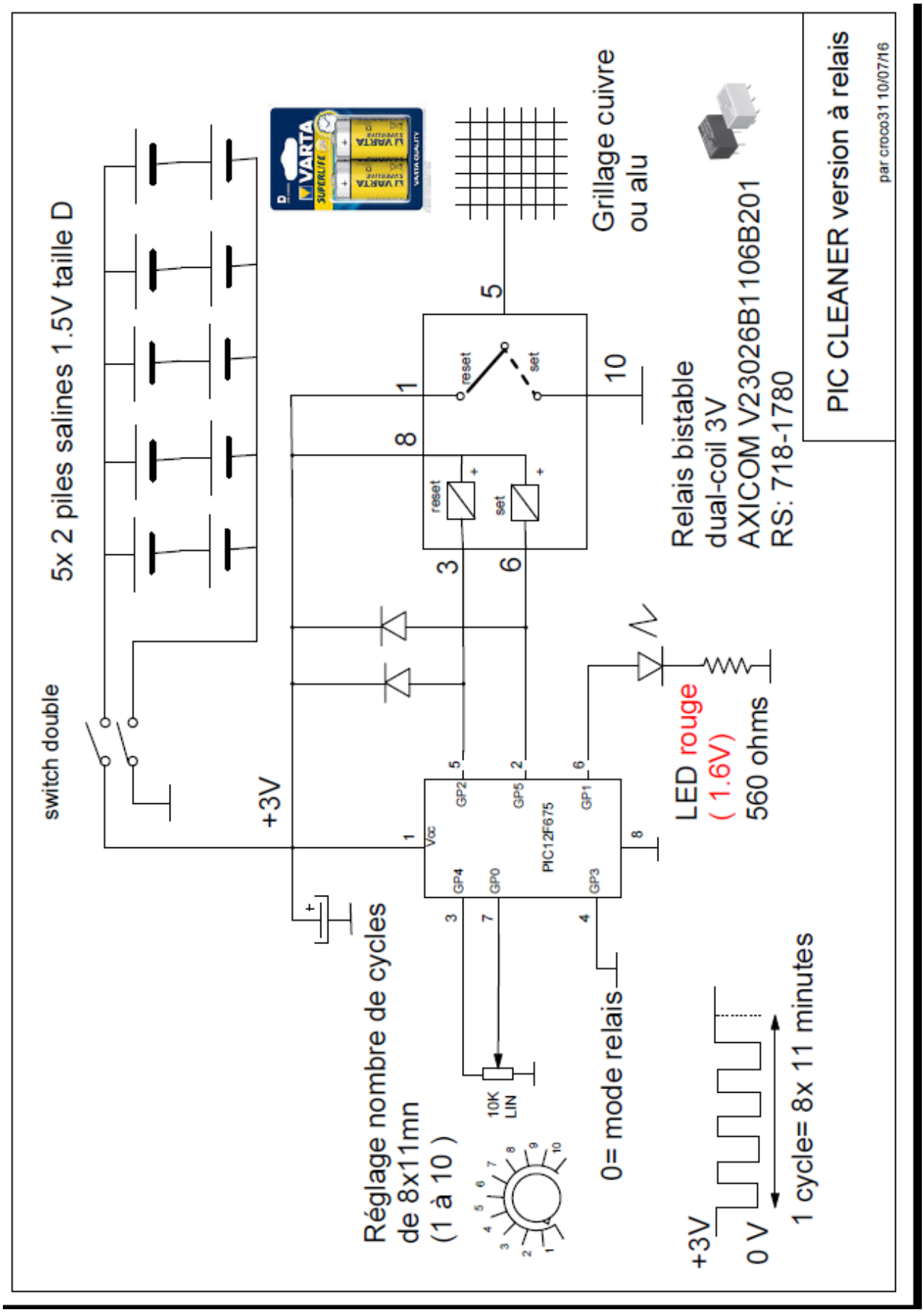
Quand tout est fini, la sortie de la grille reste sur la polarité +3V, et un flash plus fort est affiché toutes les 5 secondes environ pour indiquer de ne pas oublier de couper l'appareil (le matin normalement).

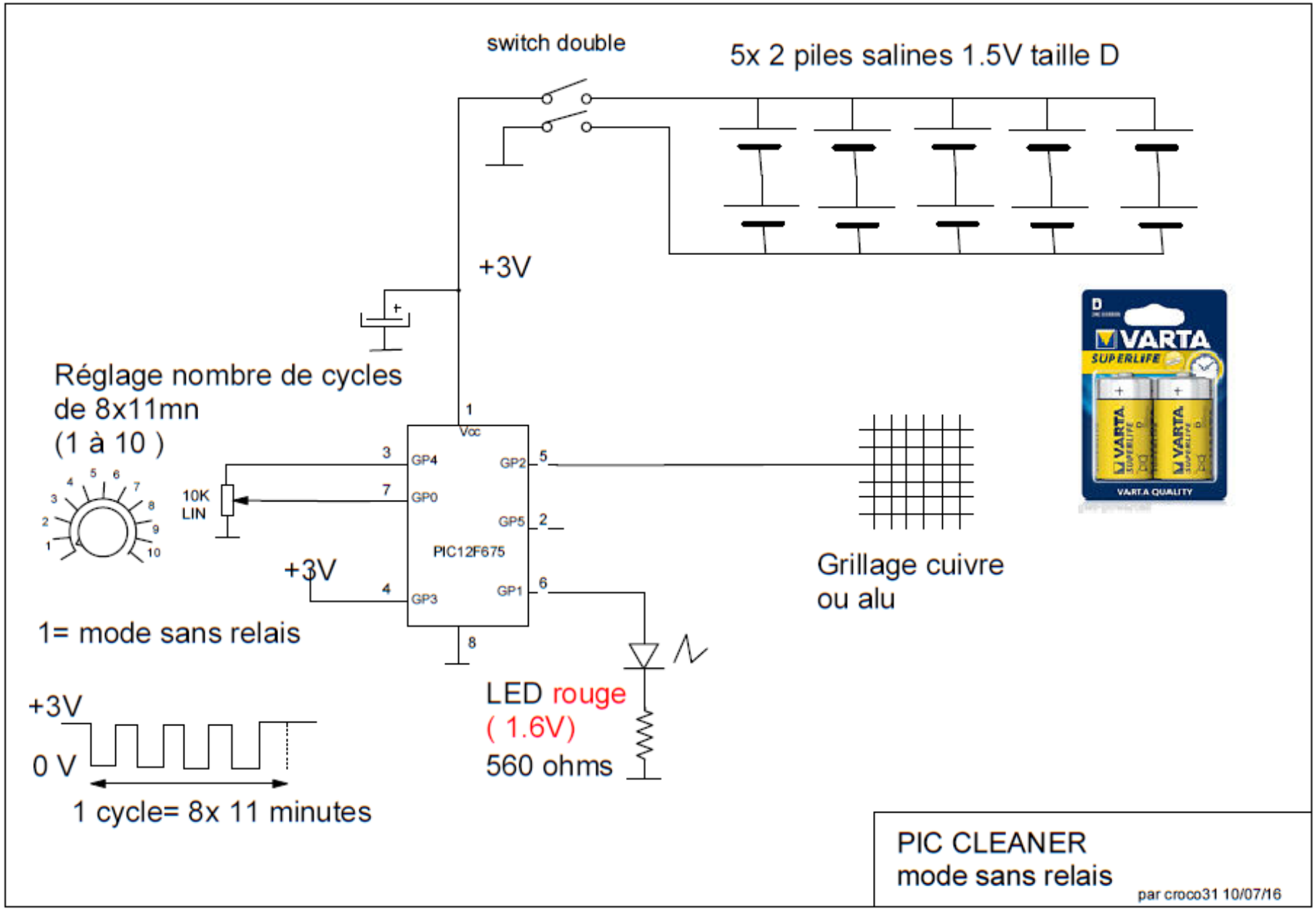
Quand on tourne le potentiomètre pendant le fonctionnement et que le logiciel détecte un changement de plage de réglage, il affiche le nombre de cycles programmé et réinitialise le processus. Il est donc conseillé de laisser le potentiomètre au milieu d'une plage de réglage.

Quand le nombre de cycles programmé est atteint, le PIC se met en mode économie d'énergie et le potentiomètre n'a plus d'action, il faut couper le switch pour recommencer une session.

7 Annexes

Schémas :




switch double
5x 2 piles salines 1.5V taille D
+3V
Réglage nombre de cycles
de 8x11mn
(1 à 10)
1
Vcc
3
GP4
GP2
5
7
GP0
10K
LIN
GP5
2
PIC12F675
+3V
4
GP3
GP1
6
8
1= mode sans relais
Grillage cuivre
ou alu
LED rouge
(1.6V)
560 ohms
+3V
0 V
1 cycle= 8x 11 minutes
PIC CLEANER
mode sans relais
par croco31 10/07/16

5

8 Code source

Fichier .H:

```
/******************************************************************************/
/* User Level #define Macros                                                  */
/******************************************************************************/

/* TODO Application specific user parameters used in user.c may go here */

// duree d'un cycle 11 minutes en secondes
#define DUREE_CYCLE_11_MINUTES 660
// nombre de cycles de 11 minutes par session = 88 minutes
#define NB_CYCLES_11MINUTES_CYCLE 8

// entree de selection de maode par pullup ou pulldown
// input only
#define pSTRAP GPIO3

// commande relais bistable bobine ON
// commande directe du relais via transistor si mise a 0V par pulldown au startup
// sortie directe en mode sans relais
#define pRELAIS_PLUS GPIO2
#define dirRELAIS_PLUS TRISIO2

// commande relais bistable bobine OFF
// commande relais monostable en mode sans relais
#define pRELAIS_MOINS GPIO5
#define dirRELAIS_MOINS TRISIO5

// sortie tension commandee pour le potar de selection
// evite de consommer par le potar de 10Kohms
#define pVPOTAR GPIO4
#define dirVPOTAR TRISIO4

// sortie de commande de la LED,
// sert de strap : si 1 au startup: LED ON si 0 si 0 au straup : LED ON si 1
#define pLEDOUT GPIO1
#define pLEDIN GPIO1
#define dirLED TRISIO1

#define LED_ON { if(MODE_LED_UP) { save_GPIOs.led =0;GPIO=save_GPIOs.byte;} else { save_GPIOs.led=1; GPIO=save_GPIOs.byte;}
dirLED=0;}
#define LED_OFF { dirLED=1;}

// valeur courante des ports en sortie
// en octet ou bits
// permet d'eviter les pbs de manipulation de ports car pas de LAT sur le 12F675
// garde la valeur de chaque port manipule en sortie
typedef union {
    unsigned char byte;
    struct {
        unsigned potar:1;
        unsigned led:1;
        unsigned relaisp:1;
        unsigned strap:1;
        unsigned vpotar:1;
        unsigned relaism:1;
    };
} type_save_GPIOs;

/******************************************************************************/
/* User Function Prototypes                                                   */
/******************************************************************************/

/* TODO User level functions prototypes (i.e. InitApp) go here */

extern unsigned char MODE_SANS_RELAIS;
extern unsigned char MODE_LED_UP;
extern volatile unsigned char cpt_5ms;
extern volatile unsigned int cpt_secondes;
extern unsigned char flag_seconde;
extern type_save_GPIOs save_GPIOs;
extern void InitApp(void);         /* I/O and Peripheral Initialization */
extern void Delai_ms(unsigned char duree);
```

Fichier .C :

```
/*****************************************************************************/
/* Polarcleaner.c: gestion des cycles
 *
 * Auteur: croco31
 * Date:   10/4/2016
 *
 * Modifs:
 *  -
 *
 *****************************************************************************/

#if defined(__XC)
    #include <xc.h>         /* XC8 General Include File */
#elif defined(HI_TECH_C)
    #include <htc.h>        /* HiTech General Include File */
#endif

#include <stdint.h>         /* For uint8_t definition */
#include <stdbool.h>        /* For true/false definition */

#include "polarcleaner.h"          /* User funct/params, such as InitApp */

/*****************************************************************************/
/* User Global Variable Declaration                                          */
/*****************************************************************************/
#define SYS_FREQ        4000000L
#define FCY             SYS_FREQ/4


// CONFIG
#pragma config FOSC = INTRCIO   // Oscillator Selection bits (INTOSC oscillator: I/O function on GP4/OSC2/CLKOUT pin, I/O function on
GP5/OSC1/CLKIN)
#pragma config WDTE = ON        // Watchdog Timer Enable bit (WDT disabled)
#pragma config PWRTE = ON       // Power-Up Timer Enable bit (PWRT enabled)
#pragma config MCLRE = OFF      // GP3/MCLR pin function select (GP3/MCLR pin function is digital I/O, MCLR internally tied to VDD)
#pragma config BOREN = ON       // Brown-out Detect Enable bit (BOD disabled)
#pragma config CP = OFF         // Code Protection bit (Program Memory code protection is disabled)
#pragma config CPD = OFF        // Data Code Protection bit (Data memory code protection is disabled)



/* i.e. uint8_t <variable_name>; */

unsigned char MODE_SANS_RELAIS;
unsigned char MODE_LED_UP;
unsigned char flag_seconde;
// valeur courante des ports en sortie
type_save_GPIOs save_GPIOs;

// nombre de cycles de 11 minutes de la seance
unsigned char nb_cycles_11minutes;
// nombre de cycles courant de 0 à nb_cycles_11minutes
unsigned char cpt_cycles_11minutes;

// gestion du potentiometre selecteur de nombre de cycles  (10 intervalles)
unsigned char val_potar,save_val_potar1,save_val_potar2;
#define NB_SEUILS 10
const unsigned char table_seuils[NB_SEUILS]= {25, 50,75,100,125,150,175,200,225,255 };
// choix du nombre de cycles de 88 minutes: 4 cycles -/+ de 11 minutes = 88 minutes
const unsigned char nb_cycles[NB_SEUILS]= { 1,2,3,4,5,6,7,8,9,10};
unsigned char num_intervalle, save_num_intervalle;
unsigned char etat_cycle;// automate de gestion

volatile unsigned char cpt_5ms;
volatile unsigned int cpt_secondes;

//******************************************************************
// Gestion interruption
//******************************************************************
void interrupt isr(void)
{
    unsigned int timer1;

    /* Determine which flag generated the interrupt */
    if(TMR1IF)
    {
     // recharge le Timer1 qui rebouclera a 0xFFFF
     // prescaler = 8 soit 8µs par pas a 4MHz
```

```
    // 5ms par tick
    // ATTENTION: corrige ici de la duree de traitement de l'interruption
    timer1= TMR1; // ici TMR1 a depasse 0000 a cause du temps de reponse de l'interruption
    TMR1= 60535 + timer1; // compte 5000 ticks de 1µs
    // compteur de secondes
    cpt_5ms++;
    if( cpt_5ms== 200) {
        cpt_5ms=0;
        cpt_secondes++; // compte les secondes
        flag_seconde=1;
    }


    TMR1IF=0; /* Clear Interrupt Flag 1 */
   }

}

// ***************************************************************************
void InitApp(void)
{
   unsigned char i;
   /* TODO Initialize User Ports/Peripherals/Project here */
   dirLED= 1; // input
   WPU= 0; // pas de weak pullups (actifs par defaut au reset)
   nGPPU= 1; // global WPU disabled (redondant: fait par defaut au reset)
   ANSEL= 0; // tous les ports en digital par defaut

   /* Initialize peripherals */
   // timer 0: sert a faire des delais precis en ms + WDT active
   // doit etre initilaise avant d'utiliser Delai_ms())
   OPTION_REG= 0b10001111; // WPUOFF, FOSC/4, prescaler sur WDT prescaler= 128
   TMR0IF=0;

   // timer1: ticks periodiques  de 5ms
   TMR1=0;
   TMR1IF=0;
   T1CON= 0b00000001; // prescaler=1 FOSC/4= 1µs TmR1ON

   // sauve l'etat courant des GPIOs en sortie pour eviter les pbs de temps de montee
   save_GPIOs.byte = 0;

   // detection du mode de fonctionnement
   MODE_SANS_RELAIS= 0;
   MODE_LED_UP= 0;
   Delai_ms(100);
   if( pSTRAP) MODE_SANS_RELAIS=1;

   /* Setup analog functionality and port direction */
   // ADC sur AN0
   TRISIO0= 1;
   ANSEL= 0b00110001; // AN0= analog , internal FRC , autres ports en digital
   ADCON0= 0b00000001; // Left justified, Vref= Vdd, channel= AN0, ADC=ON
   WPU0=0; // pas de pullup

   // commandes de relais suivant le mode choisi
   // les sorties se manipulent via save_GPIOs
   save_GPIOs.vpotar=1; // tension pour le potar
   dirVPOTAR=0;

   GPIO= save_GPIOs.byte; // met a jour les sorties

   /* Enable interrupts */
   cpt_5ms=0; // defini dans interrupts.c
   cpt_secondes=0;
   flag_seconde=0;
   PEIE=1;
   GIE=1;
   TMR1IE=1;
}

/******************************************************************************/
/* Main Program                                                               */
/******************************************************************************/

void Delai_ms(unsigned char duree)
{
   while( duree--) {
      asm("clrwdt");
      TMR0IF= 0;
      TMR0= 6; // 250µs
```

```
        while(!TMR0IF) continue;
        TMR0IF= 0;
        TMR0= 6; // 250µs
        while(!TMR0IF) continue;
        TMR0IF= 0;
        TMR0= 6; // 250µs
        while(!TMR0IF) continue;
        TMR0IF= 0;
        TMR0= 6; // 250µs  compense
        while(!TMR0IF) continue;
    }
}

// active la sortie ou le relais + / - suivant le mode choisi par les straps
// chaque cycle fait 11 minutes à - et 11 minutes à + et dure 88 minutes
void
Set_polarite_plus(void)
{
    unsigned char i;
   // si direct: etat sur la broche du PIC
    if( MODE_SANS_RELAIS) {
       // mode sans relais ou relais monostable a état permanent
       // broche 5 = etat de la grille 0 ou +3V
       // broche 2= commande eventuelle de relais inversée
        save_GPIOs.relaisp= 1;
        save_GPIOs.relaism= 0;
        GPIO= save_GPIOs.byte;
        dirRELAIS_PLUS =  0; // confirme les sorties
        dirRELAIS_MOINS=  0;
    } else {
        // relais bistable double bobine : commande par impulsion de 10ms entre les ports
        // mise a 0 du port pRELAIS_PLUS
        save_GPIOs.relaisp= 0;
        save_GPIOs.relaism= 1;
        GPIO= save_GPIOs.byte;
        dirRELAIS_PLUS =  0; // confirme les sorties
        dirRELAIS_MOINS=  1;
        Delai_ms(10);
        dirRELAIS_PLUS =  1; // tri-state les sorties
        dirRELAIS_MOINS=  1;
    }
    // indique le changement par un pulse long  sur la LED
    LED_ON;
    Delai_ms(500);
    LED_OFF;
    Delai_ms(200);

    // suivi du nombre de cycles de 88 minutes
    for(i=0;i<nb_cycles[num_intervalle];i++) {
                LED_ON;
                Delai_ms(50);
                LED_OFF;
                Delai_ms(200);
            }
}
void
Set_polarite_moins(void)
{
    unsigned char i;
   // si direct: etat sur la broche du PIC
    if( MODE_SANS_RELAIS) {
       // mode sans relais ou relais monostable a état permanent
       // broche 5 = etat de la grille 0 ou +3V
       // broche 2= commande eventuelle de relais inversée
        save_GPIOs.relaisp= 0;
        save_GPIOs.relaism= 1;
        GPIO= save_GPIOs.byte;
        dirRELAIS_PLUS =  0; // confirme les sorties
        dirRELAIS_MOINS=  0;
    } else {
        // relais bistable double bobine  commande par impulsion de 10ms entre les ports
        // pulse positif entre broche 5 et broche  2
        save_GPIOs.relaisp= 1;
        save_GPIOs.relaism= 0;
        GPIO= save_GPIOs.byte;
        dirRELAIS_PLUS =  1 ; // confirme les sorties
        dirRELAIS_MOINS=  0;
        Delai_ms(10);
        dirRELAIS_PLUS =  1; // tri-state les sorties
        dirRELAIS_MOINS=  1;
    }
```

```
    // indique le changement par un pulse moyen  sur la LED
    LED_ON;
    Delai_ms(200);
    LED_OFF;
    Delai_ms(400);

    // suivi du nombre de cycles de 88 minutes
    for(i=0;i<nb_cycles[num_intervalle];i++) {
            LED_ON;
            Delai_ms(50);
            LED_OFF;
            Delai_ms(200);
        }
}
// Redemarre une session de nb cycles de 88 minutes = 4 cycles -/+ de 2x11 minutes
void
Demarre_session(unsigned char nb)
{

    Set_polarite_moins();
    etat_cycle= 0; // init machine a etats  et cycle - en cours
    nb_cycles_11minutes= nb*NB_CYCLES_11MINUTES_CYCLE; // 1 cycle = 88 minutes = 4x cycle-/cycle+
    cpt_cycles_11minutes= 0;
    PEIE=0;
    cpt_5ms= 0;
    cpt_secondes=0;
    TMR1IF= 0;
    PEIE=1;
}
// automate de gestion des cycles
unsigned char
Automate_cycles(void)
{
    unsigned int cpt_secondes_courant;
    // lecture masquee du compteur de secondes
    PEIE=0;
    cpt_secondes_courant= cpt_secondes;
    PEIE=1;
    if( cpt_secondes_courant >= DUREE_CYCLE_11_MINUTES) {
        PEIE=0;
        cpt_secondes=0;// gere par interupt
        PEIE=1;
        // nouveau cycle de 11 minutes est passe
        cpt_cycles_11minutes++;
         if(cpt_cycles_11minutes>= nb_cycles_11minutes) {
            // termine: on sort
            return 1;
         }
        // gestion des basculement de relais
        switch(etat_cycle) {
            default:
            case 0: // on etait en cycle negatif ou depart
                Set_polarite_plus();
                etat_cycle= 1;
                break;
            case 1: // on etait en cycle positif
                Set_polarite_moins();
                etat_cycle= 0;
                break;

        }
    }


    return 0;
}

// arret session: mise en standby
void
Arret_session(void)
{
    unsigned char i;
   // arret des commandes relais suivant le mode
   // on termine en polarite plus
   Set_polarite_plus();

   // indique la fin par une serie de 10 pulses sur la LED
   for(i=0;i<10;i++) {
        LED_ON;
        Delai_ms(200);
        LED_OFF;
```

```
    Delai_ms(200);
  }

  // stop ADC pour economiser le courant
  ADON=0;

  // stop LED
  LED_OFF;

  // stop de la tension du potar
  save_GPIOs.vpotar=0;
  GPIO= save_GPIOs.byte;

  // stop Timer1
  TMR1ON= 0;

  // SLEEP avec reveil periodique par le WDT
  // le WDT reactive en sequence
  while(1) {
    GIE=0; // disable interrupts
    asm("SLEEP");
    // pulse periodique sur la LED pour indiquer l'etat ON
    LED_ON;
    GIE=1;
    Delai_ms(200);
    LED_OFF;
  }

}
// test de la valeur du potar et calcul du nombre de cycles programme
void test_potar(void)
{
   unsigned char i,seuil1,seuil2,num;
   ADCON0bits.GO=1;
      while(ADCON0bits.GO) continue;
      val_potar= ADRESH; // de 0 a 255
      // voir si le potar a change d'intervalle
      // calcul de l'intervalle courant de selection de 0 a 9
      // avec protection par fenetre autour du seuil de la table
      seuil1= table_seuils[num_intervalle]; // intervalle courant
      // on ne bouge pas autour de l'intervalle courant
      if( seuil1!=255) {
            // pas le dernier
            seuil1= seuil1-2;
            seuil2= seuil1+2;
      } else {
            // le seuil haut maxi 255
            seuil1=seuil1-2;
            seuil2=255;
      }
      if( (val_potar>seuil1) && (val_potar<seuil2) ) return;
      // le potar est sorti de l'intervalle courant
      for(i=0;i<NB_SEUILS;i++) {
         seuil1= table_seuils[i];
         if( val_potar <= seuil1) {
            // valeur dans l'intervalle i, i+1
            num=i;
            break;
         }
      }

      num_intervalle= num;
}
void main(void)
{
   unsigned char i,flag_start;

   asm("clrwdt");
   /* Initialize I/O and Peripherals for application */
   InitApp();

   // clignote au startup
   for(i=0;i<10;i++) {
      LED_ON;
      Delai_ms(50);
      LED_OFF;
      Delai_ms(100);
   }

   Set_polarite_plus(); // position RESET du relais par defaut
```

```
    flag_start=0;
    // tempo de 1 secondes
    flag_seconde=0;
    cpt_5ms=0;
    while(!flag_seconde) continue;


    num_intervalle=0;
    save_num_intervalle= 255; // force un changement initial
    while(1)
    {
      /* TODO <INSERT USER APPLICATION CODE HERE> */
      // lecture du potentiometre pour selecter le nombre de cycles de 3x11 minutes de 1 a 10
      // chaque franchissement de seuil remet a 0 la duree de timing
      asm("clrwdt");

      // test de la valeur du potar et calcul du nombre de cycles
      test_potar();

      // teste si changement de nombre de cycles par la position du potar (qui a bouge))
      if( num_intervalle != save_num_intervalle) {
        // premiere selection apres startup ou modif de la selection par le potar
        save_num_intervalle= num_intervalle;
        // affiche le choix par un nombre de pulses correspondants sur la LED
        for(i=0;i<nb_cycles[num_intervalle];i++) {
          LED_ON;
          Delai_ms(50);
          LED_OFF;
          Delai_ms(200);
        }
        cpt_5ms=0;
        cpt_secondes=0; // pour la tempo de 2 secondes
        flag_start=0;// relance un cycle si le nombre de cycles a change
      } else {
        // relance un cycle au bout de 5 secondes de stabilite du potar
        if( (cpt_secondes>=5 )&& (!flag_start) ) {
         Demarre_session(nb_cycles[num_intervalle]);
         flag_start=1;
        }
      }

     // appel periodique de l'automate de gestion d'avancement des cycles
      // renvoie 1 si fin des cycles (passage en standby)
     if( flag_start) {
           // indique que cela tourne par un pulse bref sur la LED
           // chaque seconde
        if(flag_seconde) {
          flag_seconde=0;
          LED_ON;
          Delai_ms(5);
          LED_OFF;
        }
       if( Automate_cycles() ) {
       // arret de la session par mise en standby sans consommation
       // on en sort un reset suite a mise OFF /ON
       // impulsion periodique sur la LED pour signaler l'etat ON
        Arret_session();
       }
     }
    }

}
```

9 Programmation du PIC

Fichier .HEX :

```
:020000000F28C7
:10000800DE00030E8312A2000408A3000A08A4005D
:100018005F08A5001F2A8316FF2390008312142867
:10002800831331308400403 0A32283011B286400ED
:10003800BD218312B0010A30300203183C2833086E
:1000480003192828BB102928BB143B0885008316F0
:10005800851032305022831685146430502201 30C6
:100068008312AE002E08B0070A303002031C232882
```

```
:100078003A218312AF01B801B501380803194128A4
:10008800BA01FF30AE002E08BC006400A420831221
:100098003A083C06031977283A08AE002E08BC0037
:1000A800B0013A08013E84006F223002031872281A
:1000B800330803196128BB106228BB143B0885006C
:1000C80083168510323050228316851 4C83050228A
:1000D80001308312AE002E08B0075528B501B101D2
:1000E800B201AF0189280030320205300319310 20C
:1000F800031C8928AF08031D89283A08013E84009B
:100108006F2289228312AF01AF0A2F0803194928E9
:10011800380803199E28B801330803199528BB101D
:100128009628BB143B0885008316851005305022 9D
:1001380083168514EE21003A03194928822149289B
:1001480083129F149F18A6281E08A8002808BF001D
:100158003A080B3E84006F22A8002808AC002C083F
:10016800FF3A0319BF282C08FE3EA8002808AC0057
:100178002C08023EC5282C08FE3EA8002808AC0022
:10018800FF30A8002808AA003F082C020318D0282E
:100198002A083F02031C0800AB010A302B0203188F
:1001A800ED282B080B3E84006F22A8002808AC001D
:1001B8003F082C02031CE5282B08A8002808A900E2
:1001C800ED280130A8002808AB070A302B02031CD1
:1001D800D5282908A8002808BA000800831234087E
:1001E8000319FE283B11BB163B0885008316051131
:1001F80085120A293B15BB123B0885008316051595
:1002080085120A305022831605158516831233 0885
:100218000319102 9BB101129BB143B08850083164C
:100228008510C8305022831685149030502283 12CE
:10023800AA013A08013E84006F222A020318080026
:1002480033080319292 9BB102A29BB143B08850048
:1002580083168510323050228316851 4C8305022F8
:100268000130 8312A9002908AA071D29831234081E
:1002780031946293B15BB123B0885008316051157
:1002880085125229 3B11BB163B08850083160511C0
:100298008516 0A305022831605158516831233 08F1
:1002A80003195829BB105929BB143B08850083162C
:1002B8008510F430502283168514C83050228312DA
:1002C800AA013A08013E84006F222A020318080096
:1002D800330803197129BB107229BB143B08850028
:1002E80083168510323050228316851 4C830502268
:1002F80001308312A9002908AA0765293A21831227
:10030800AC010A302C020318A229330803198E29DC
:10031800BB108F29BB143B08850083168510C83095
:1003280050228316851 4C830502201308312AB0046
:100338002B08AC070A302C02031C89291F108316CE
:1003480085148312 3B123B08850010108B13630041
:1003580083123308031 9B229BB10B329BB143B0815
:1003680085008316851 08B17C83050228316851494
:10037800AA298316851495018117 9F018F30810062
:100388000B1183128E018F010C1001309000BB01FC
:10039800B401B3016430502283128 51DD529B401FC
:1003A800B40A8316051431309F00013083129F0070
:1003B8008316151083123B168316051283123B0809
:1003C8008500B501B101B201B8010B178B1783166F
:1003D8000C1408000B1383123208AD01AD07310865
:1003E800AC01AC070B1702302D02943003192C0214
:1003F800031C1D2A0B13B101B2010B170130AB000E
:100408002B08B60739083602031C152A01300800E4
:100418003A218312B701B70A1D2AF2208312B701C5
:100428001D2A3708003A03190C2A013A0319112A20
:100438000C2A003008000C1C452A0F08A701A70742
:100448000E08A601A6072608773EA000270803186D
:10045800013EEC3EA10020088E0021088F000130EB
:10046800A0002008B5073508C83A031D442AB5017D
:100478000130B1070318B20A0030B207B801B80A50
```

```
:100488000C102508DF0024088A0023088400220EA7
:100498008300DE0E5E0E09008312A8000130A80258
:1004A8002808FF3A0319080064000B11063083126C
:1004B80081000B1D5D2A0B11063081000B1D622A7D
:1004C8000B11063081000B1D672A0B1106308100C5
:1004D8000B19522A6C2A02308A000408840A8207FF
:1004E800003401340234033404340534063407344 8
:1004F800083409340A34193432344B3464347D34C2
:100508009634AF34C834E134FF348312AD00F2209E
:100518008312B7012D08AB0002300310AB0DFF3E6C
:10052800031D912A03102B0DAC002C08B900B6014D
:100538000B13B501B101B2010C100B1708006400D0
:0E0548008001840A04060319003404 06A42A64
:02400E00CC31B3
:00000001FF
```