Le générateur d'argent colloidal du crocodile
Façon Spooky2

Version : V1.0
Date :1 mars 2021
Auteur : croco31

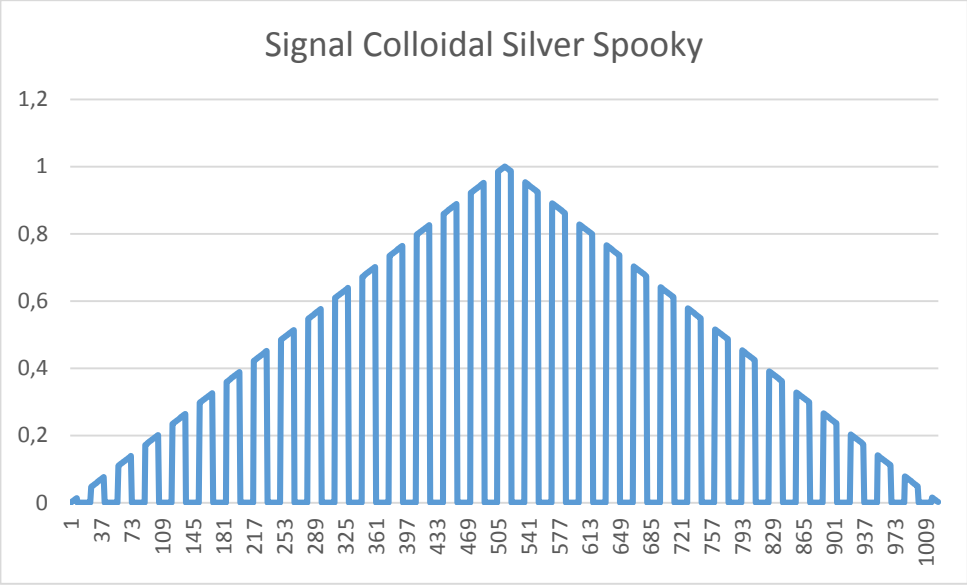
Résumé :
Ce document décrit la construction d'un générateur d'argent colloïdal basé sur le signal utilisé par Spooky2. Ce signal a la vertu de générer des colloïdes d'argent de très faible dimension nanométrique mais demande une longue durée de fabrication. Le générateur est intégré dans un agitateur magnétique, qui reste toujours utilisable en dehors de la fabrication de l'argent colloïdal, e profitant en plus d'une temporisation de l'agitateur.

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

1. Introduction

Le générateur d'argent colloïdal Spooky2 est basé sur l'utilisation d'un générateur XM ou GX, générant un signal particulier qui a la propriété de fabriquer des colloïdes d'argent de très faible dimension nanométrique, ayant des propriétés dites plus efficaces au niveau thérapeutique car plus facilement assimilées. Le signal choisi évite de fabriquer des ions d'argent AG++ qui réagissent avec les chlorures de l'eau.
La méthode est décrite sur le site Spooky2 : https://www.spooky2.fr/

Le signal utilisé à 5Hz est basé sur une tension triangulaire de 18V, découpée périodiquement et affectée d'un léger offset.



Une résistance de 10Kohms est placée en série avec les électrodes d'argent (diamètre 2 à 3mm pour avoir une bonne surface de contact) pour limiter le courant à une très faible valeur (0.2mA environ).
L'inconvénient est que la durée de fabrication d'une solution d'argent colloïdal à 15 ou 20ppm (l'optimum) prend plusieurs dizaines d'heures, ce qui monopolise un générateur et un PC pour cet usage (le générateur GX – onéreux- peut s'affranchir du PC).

L'objectif ici est de recréer ce signal à partir d'un circuit simple et peu coûteux.

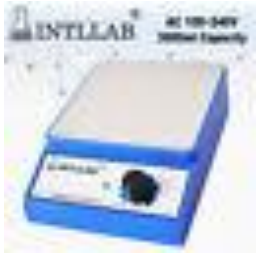
2. Fonctions à réaliser

Les fonctions suivantes doivent être effectuées :

- Génération du signal triangulaire type Spooky d'amplitude 20V environ et affecté d'un offset de 2V environ (10%)
- Inversion du sens du signal toutes les 4 minutes pour ne pas oxyder les électrodes (en pratique elles s'oxydent quand même)
- Brassage périodique de la solution
- Arrêt automatique au bout du temps programmé

La durée sera réglable dans une plage de 1 à 100heures environ, ce qui est suffisant.

Le générateur sera intégré directement dans un agitateur magnétique existant afin d'éviter le recours à un boitier séparé et réduire le coût.
L'agitateur utilisé est celui d'Intllab trouvé sur ebay pour 33^{E} :
https://www.ebay.fr/itm/Magnetic-Stirrer-Laboratory-Lab-INTLLAB-Mixer-With-3000rpm-AC-100V-240V-New/253899777921?hash=item3b1d9b2781:g:bwEAAOSwv~NZs1bp



Cet agitateur est livré avec son alimentation 9V et un aimant enrobé pour l'agitation du liquide.

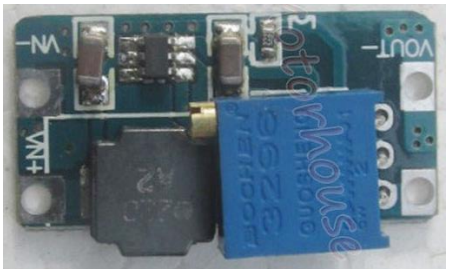
3. Choix du circuit et du µP

Pour pouvoir fabriquer facilement le signal et gérer la durée, en minimisant le nombre composants, le circuit est basé sur un µP PIC16F1709 associé à un AOP double LF412.
Ce µP est intéressant car il comporte :

- Un oscillateur interne à 32MHz calibré en usine à 1%
- Un DAC 8 bits permettant de générer un signal analogique
- Des timers pour cadencer le signal et gérer la durée
- Un ADC pour capture la position du potentiomètre de réglage

Le µP fonctionne en +5V, obtenu à partir de l'alimentation 9V de l'agitateur via une régulateur low-drop LM78L05.

La tension 25V nécessaire à l'ampli op LF412 et pour obtenir l'amplitude du signal, est obtenue avec un module Step-Up qui élève la tension 9V à 25V :



Disponible sur ebay : https://www.ebay.fr/itm/DC-DC-Converter-Boost-Step-up-Voltage-Mini-Module-3V-5V-9V-12V-24V-2A-Adjustable/192390529656?hash=item2ccb5e8278:g:8aYAAOSwGPxaKgbn

Au repos les sorties des électrodes sont à +25V /masse (AOP saturé) ce qui donne une différence de tension nulle entre les électrodes. Ceci est dû au fait que LF412 n'accepte pas de tension trop basse sur ses entrées + et -.Il fonctionne donc à l'envers pour le découpage du signal (signal coupé= +25V).

Voir schéma en annexe.

4. Réglage de durée et démarrage

Afin de simplifier le montage, seul un potentiomètre est utilisé pour régler la durée et démarrer/arrêter une fabrication.
Une LED indique l'état de fonctionnement :

LED	Etat
Clignote à 1Hz 0.5s ON 0.5 OFF	En attente, le moteur peut tourner pour régler sa vitesse, pas de signal sur les électrodes
LED allumée fixe	Signal actif, le moteur n'est activé que pendant 5 secondes à chaque minute. Sa vitesse peut être ajustée par le potentiomètre de l'agitateur
LED clignotement bref et rapide	Durée atteinte Le moteur est inactif pour laisser décanter la solution.

Pour démarrer une fabrication, le potentiomètre doit effectuer la séquence suivante :

- Potentiomètre au maximum pendant 1 seconde environ
- Potentiomètre au minium pendant 1 seconde environ
- Potentiomètre à la position correspondant à la durée choisie.

Le démarrage est effectif quand la LED s'allume fixe.

Il est possible d'ajuster la durée après le démarrage en tournant le potentiomètre.

Le démarrage peut être annulé en mettant le potentiomètre en position minimale : la LED se remet à clignoter.

La plage de réglage de durée va de 60 à 5600 minutes soit 4 jours environ.

L'usage d'un potentiomètre linéaire permet de marquer grossièrement des index de durée, mais un strap est prévu pour marquer précisément ces index.

5. Calibration du potentiomètre

Cela demande un oscilloscope.
Pour cela il faut activer le strap de calibration (à la masse) qui provoque les choses suivantes :

- La LED s'allume chaque seconde avec une durée proportionnelle à la durée réglée sur le potentiomètre
- Un signal carré de durée proportionnelle à la durée finale est émis chaque seconde sur l'électrode :
  Correspondance : 0.1ms par minute réglée

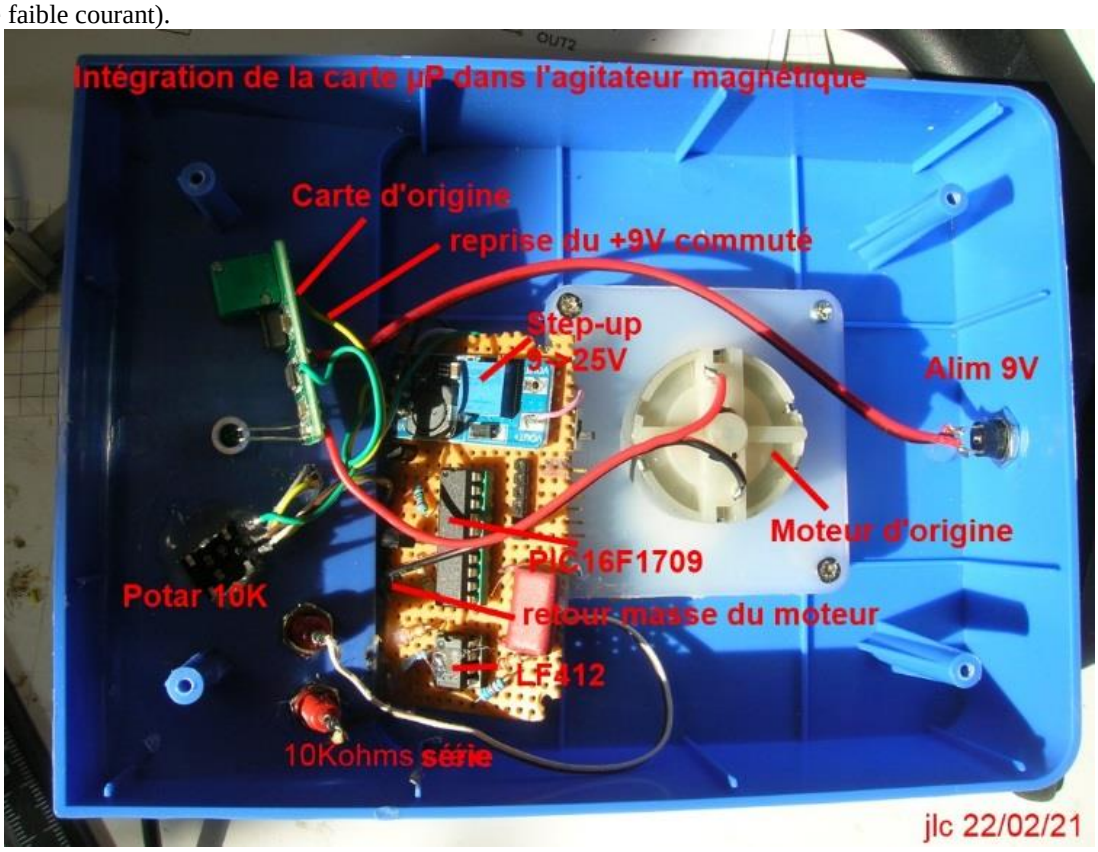
Il suffit de mesurer la durée de cette impulsion pour connaître la durée en minutes qui correspondra à la position courante du potentiomètre. Un marquage au feutre suffit à repérer chaque dizaine d'heures par exemple.

Ne pas oublier d'ôter le strap pour avoir la fonctionnement normal.

Comme le potentiomètre est linéaire, en repérant les positions extrêmes, il est possible de faire une proportion pour le marquage. La précision sur la durée n'est pas critique.

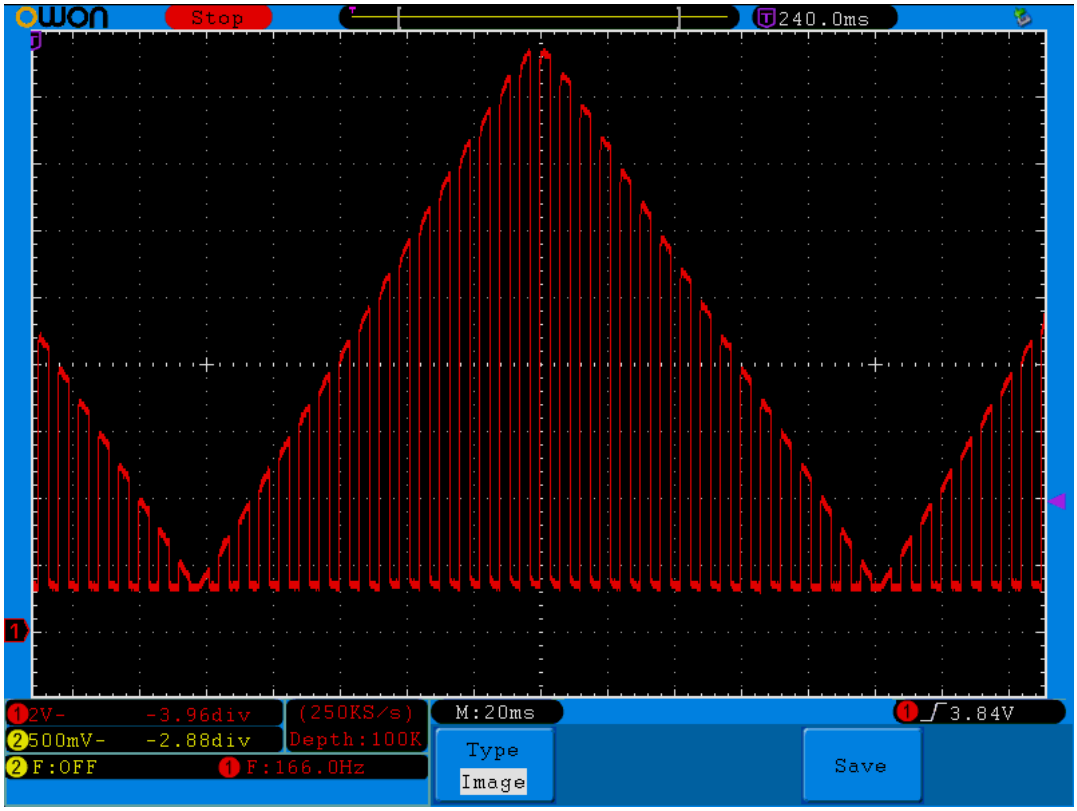
6. La réalisation

Vu le faible nombre de composants le circuit se monte facilement sur une carte prototype, collée à la colle chaude l'intérieur du boitier de l'agitateur. La tension 9V est reprise sur la carte d'origine en aval du switch du bouton de réglage du moteur. De même le fil – du moteur est repris vers la nouvelle carte pour assurer sa commutation par le µP via le transistor (un NPN quelconque est suffisant vu le faible courant).



Intégration de la carte dans le boitier de l'agitateur

Le générateur terminé

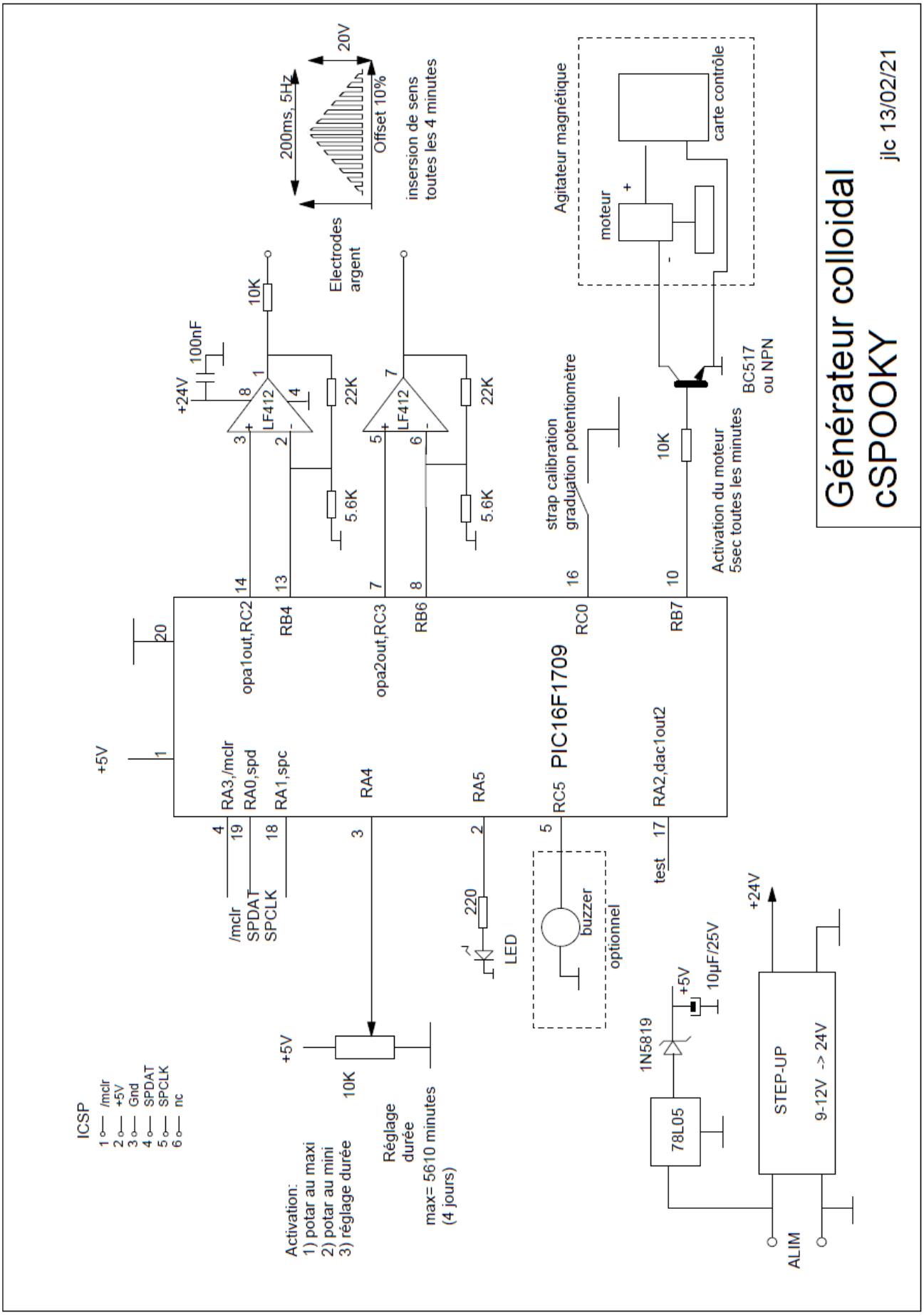


Le signal différentiel entre les électrodes

Programmation du µP

Le fichier .HEX est copié en annexe. Il suffit de copier le code hexadécimal dans un fichier texte et de le programmer sur un PIC16F1709 avec un programmateur adéquat.

Schéma : note : la diode 1N5819 ne sert qu'à séparer le +5V du programmateur ICSP Pickit3



nota
Source du programme :
Compilé sur la chaîne gratuite MPLAB IDE sous XC

Programme include : colloider.h

```
/*
 * File:   colloider.h
 * Author: croco31
 *
 */

#ifndef MINISPOOKY_H
#define MINISPOOKY_H

#ifdef  __cplusplus
extern "C" {
#endif

#include <xc.h>

/* Definition des ports utilises sur le PIC16F1709 */
/* ------------------------------------------------*/


// commande du buzzer
#define pBUZZER LATC5
#define dirBUZZER TRISC5

// commande du moteur
#define pMOTOR LATB7
#define dirMOTOR TRISB7


// sorties analogiques de OPA1OUT et OPA2OUT
#define pOPA1OUT LATC2
#define pOPA2OUT LATC3
#define dirOPA1OUT TRISC2
#define dirOPA2OUT TRISC3

// entree su strap de calibration avec pullup
#define pSTRAP RC0
#define dirSTRAP TRISC0

// entree du potentiomètre de réglage de durée
#define pPOTAR RA2
#define dirPOTAR TRISA2

// commande LED
#define pLED LATA5
#define dirLED TRISA5

// init des divers ports
#define INIT_PORTS() {ANSC2=1;ANSC3=1;ANSA2=1;ANSC0=0;nWPUEN=0;WPUC0=1;TRISA5=0;LATA5=0;TRISA2=1;TRISC0=1;TRISC5=0;TRISB7=0; }

extern void Beep(void);
extern void Delai_ms(unsigned int);
extern void Delai_100us(unsigned int duree);

#ifdef  __cplusplus
}
#endif
#endif  /* MINISPOOKY_H */
```

Programme pincipal colloider.c

```
/*
 * File:   colloider.c
 * Author: croco31
 * generateur de signal spooky2 pour argent colloidal
 */

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

#include <xc.h>


// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

// CONFIG1
#pragma config FOSC = INTOSC    // Oscillator Selection Bits (INTOSC oscillator: I/O function on CLKIN pin)
#pragma config WDTE = OFF       // Watchdog Timer Enable (WDT disabled)
#pragma config PWRTE = ON       // Power-up Timer Enable (PWRT enabled)
#pragma config MCLRE = OFF      // MCLR Pin Function Select (MCLR/VPP pin function is digital input if LVP bit is also 0.)
#pragma config CP = OFF         // Flash Program Memory Code Protection (Program memory code protection is disabled)
#pragma config BOREN = ON       // Brown-out Reset Enable (Brown-out Reset enabled)
#pragma config CLKOUTEN = OFF   // Clock Out Enable (CLKOUT function is disabled. I/O or oscillator function on the CLKOUT pin)
#pragma config IESO = OFF       // Internal/External Switchover Mode (Internal/External Switchover Mode is disabled)
#pragma config FCMEN = OFF      // Fail-Safe Clock Monitor Enable (Fail-Safe Clock Monitor is disabled)

// CONFIG2
#pragma config WRT = OFF        // Flash Memory Self-Write Protection (Write protection off)
#pragma config PPS1WAY = OFF    // Peripheral Pin Select one-way control (The PPSLOCK bit can be set and cleared repeatedly by software)
#pragma config ZCDDIS = ON      // Zero-cross detect disable (Zero-cross detect circuit is disabled at POR and can be enabled with ZCDSEN bit.)
#pragma config PLLEN = OFF       //
#pragma config STVREN = ON      // Stack Overflow/Underflow Reset Enable (Stack Overflow or Underflow will cause a Reset)
#pragma config BORV = LO        // Brown-out Reset Voltage Selection (Brown-out Reset Voltage (Vbor), low trip point selected.)
#pragma config LPBOR = OFF      // Low-Power Brown Out Reset (Low-Power BOR is disabled)
#pragma config LVP = OFF        // Low-Voltage Programming Enable (High-voltage on MCLR/VPP must be used for programming)
```

```
#include "colloider.h"

// ------------------ variables globales --------------------------

volatile unsigned int cpt_4ms;
volatile unsigned int cpt_500us;
volatile unsigned char cpt_secondes;
volatile unsigned int cpt_minutes;
volatile bit flag_secondes,flag_sens_signal,flag_signal_actif;
volatile bit flag_minutes;
unsigned char val_potar;
volatile unsigned char val_signal;
volatile unsigned char cpt_hache;
volatile bit flag_hache;
unsigned int duree_max;

// la difference donnera 100ms environ
#define OFFSET_SIGNAL 50
#define MAX_SIGNAL 253

// ************************************************************************
// Interruptions
void interrupt interrupt_manage(void)
{
    // frequence de base du signal a 4ms
    if (TMR4IF ) {
        TMR4IF=0;

    } // ---------- TMR4IF

    // tick a 500µs, ce qui donne 200 points pour 100ms
    if( TMR6IF) {
        TMR6IF=0;
        if( flag_signal_actif) {
        // signal sur le DAC: ici pour eviter de la gigue
            if( flag_hache) {
                DAC1CON1= MAX_SIGNAL;
            } else {
                DAC1CON1= val_signal;
            }

        if( flag_sens_signal) {
         // signal montant pendant 100ms
            val_signal++; //signal triangulaire non hache
            if( val_signal>MAX_SIGNAL) {
                // redescend
                flag_sens_signal=0;
            }
        } else {
         // signal descendant pendant 100ms
            val_signal--;
            if( val_signal<=OFFSET_SIGNAL) {
                flag_sens_signal=1; // remonte
            }
        }
            // hachage du signal periode 3ms = 6 ticks
            cpt_hache++;
            if( cpt_hache>5) {
                cpt_hache=0;
                flag_hache = !flag_hache; // passe / passe pas
            }
        }
        cpt_500us++;
        if( cpt_500us>=2000) {
            cpt_500us=0;
            cpt_secondes++;
            flag_secondes=1;
            if( cpt_secondes>=60) {
                cpt_secondes=0;
                cpt_minutes++;
                flag_minutes=1;
            }
        }

    }
}
// ************************************************************************


// ------------- acces a la flash endurante de 128 bytes en 0x1F80 à 0x1FFF ------------------
//
// bloc d'infos de configuration (max 32 bytes)

 struct {
    unsigned char tag; // tag de marquage de validite de la configuration stockee (0x55))
    unsigned char cal_osc;// calibration oscillateur interne

 } Configuration_flash;

// lecture des octets depuis le ROW#0
 // 0 si OK -1 si erreur de parametre
char  Read_flash_configuration(void)
{
    return HEFLASH_readBlock( (void *)&Configuration_flash,0,sizeof(Configuration_flash));
}
// ecriture du buffer local de configuration en flash endurante
 // 0 si OK -1 si erreur de parametre  1 si erreur d'ecriture
char Write_flash_configuration(void)
{
    return HEFLASH_writeBlock(0,(void *) &Configuration_flash,sizeof(Configuration_flash)); // write
}

void ConfigureOscillator(void)
{
    /* Activation INTOSC a 32MHz PLLx4= ON  */
    OSCCON= 0b11110000; /* SPLLEN=1 IRCF=1110 SCS=00*/
```

```
    // attend que la PLL soit stable
    while( !PLLR) continue;
    // attend que l'oscillateur HFINTOSC soit stable
    while( !HFIOFS) continue;
}

/* Activation des timers */
void ConfigureTimers(void)
{
   /* Freq/4 de base = 8MHz */
   /* TIMER0 utilise pour generer un Tick sur TMR0IF et ITs eventuelle */
   /* Prescaler = 1:256 soit 32uS en entree */
   /* Timer0 reboucle en 32x256uS soit un tick de 8ms environ */
    /* sert a declencher l'ADC en mode trigger */
    TMR0CS= 0; /* Source = Freq/4 */
    TMR0IF= 0;
    PSA= 0; /* Prescaler */
    PS2 = 1; /* PS = 111 pour 1:256 */
    PS1 = 1;
    PS0 = 1;
    TMR0IE=0; // pas d'IT utilisee sur Timer0



    /* Timer1 utilise comme horloge de synchro du COMP1 a 1µs */
    /* source FOSC/4 = 8MHz prsescaler= 1:8 non synchronise avec FOSC/2*/
    T1CON= 0b00110101;
    // T1 utilise comme timer pour le delai 100µs sur modif de TMR1

    /* */
    /* Timer2 utilise comme reference pour le PWM3 et PWM4 */
    /* Periode= Tosc*4 * (PR2+1) * Prescaler */
    /* si prescaler=4 clock= 2MHz Periode= 0.5µs * (PR2+1) */
    PR2= 39; // periode T2MATCH= 20µs 50KHz
    T2CON= 0b00000101; // prescaler= 4 postscaler=1 F= 2MHz
    // Freq NCO= 0.5* Increment/2*1048576 MHz = Increment/4.194304 Hz
    TMR2IE=0;




    /* Timer4 est utilise comme tick a 10ms sur TMR4IF pour scruter les switches */
    /* utilise Fosc/4 = 8MHz Prescaler = 1/64 PR2=125 Postscaler= 10 */
    /* PR= 1ms de base utilise comme periode pour le PWM3 */
    PR4=124;
    TMR4IF=0;
    T4CON= 0b01001111;
    TMR4IE= 0;


    // Timer6 est utilise comme tick de 500µs
    /* utilise Fosc/4=8MHz  Prescaler = 1/16 PR6=250 Postscaler= 1 */
    // freq= 1/8MHz= 2µs
    PR6= 249;// PR6+1= 500µs
    TMR6IF=0;
    T6CON= 0b00000110; // prescaler=1/16 postscaler=1
    TMR6IE= 1; //
    val_signal=OFFSET_SIGNAL;
    flag_hache=0;
    cpt_500us=0;
    cpt_secondes=0;
    cpt_minutes=0;


    // ADC pour lecture du potar
    /* ---------------------------ADC ------------------------------------- */
    /* Activation de l'ADC en mode trigger par Timer0 overflow */
    /* VREF+= Vdd */
    /* Source= AN3 port RA4 broche 3*/
    /* ADC clock = FRC interne */
    ADCON0= 0b00001101; /* AN3 GO=0 ADC_ENABLE=1*/
    ADCON1= 0b00110000; /* Left justified, FRC , VREF+= Vdd */
    ADCON2= 0b00110000; /* trigger automatique : ADC active par overflow du Timer0 a 8ms */
    ANSA2= 1; // analog input RA2
    TRISA2=1;

    // OPA1 OPA2 actifs sur les sorties
    //OPA1CON= 0b11010010; // sur DAC
    //OPA2CON= 0b11010010; // sur DAC

    // DAC 8 bits
    DAC1CON0= 0b10011000; // Vref= sortie FVR  DAC1OUT2 sort sur broche RA2
    DAC1CON1= 0;

    // FVR
    FVRCON= 0b10001100; // 4.096V pour DAC

}


// Delai par ticks de 100µS (Timer1 a 1µs)
void Delai_100us(unsigned int duree)
{

    while(duree !=0) {

        TMR1= 65435; // TMR1IF monte sur le roll 0xFFFF->0x0000
        TMR1IF=0;
        while(!TMR1IF) continue;
        duree--;
    }
}

// Delai en milliseondes
void Delai_ms(unsigned int duree)
{
    Delai_100us(duree*10);
}
```

```
// beep sur le buzzer
void Beep(void) {
    unsigned char i;
    for(i=0;i<25;i++) {
        pBUZZER=1;
        Delai_100us(20);
        pBUZZER=0;
        Delai_100us(20);
    }
}



// init des compteurs de temps
void Init_compteurs(void)
{
     di();
     cpt_500us=0;
     cpt_secondes=0;
     cpt_4ms=0;
     cpt_minutes=0;
     cpt_hache=0;
     ei();
}

void Init_Ports(void)
{
   /// init des ports
    INIT_PORTS();

}

void main(void)
{
    char flag_inversion,flag_start;
    unsigned int cpt_calibre,cpt_sec;

    Init_Ports();
    ConfigureOscillator();
    ConfigureTimers();



    //  ---------- active les interruptions
    PEIE= 1;
    ei(); // besoin IT 1ms pour Delai_ms()

    Init_compteurs();
    flag_start=0;

  STARTUP:
    // Calibration du potentiometre si strap actif a 0
    // TMR6IE=1 necessaire ici pour flag_secondes

    // moteur actif par defaut
    pMOTOR=1;

    if( pSTRAP==0) {
        // la LED s'allume  autant de 0.1ms que de minutes dans la duree
        goto CALIBRE;
    }

    DAC1CON1= MAX_SIGNAL;// 0 pas de signal en sortie
    // OPA1OUT , OPA2OUT mis a 0
    OPA1CON= 0;
    OPA2CON= 0;
    dirOPA1OUT=0;//
    pOPA1OUT=1;// ici les 2 sorties du LF412 seront à 20V= pas de courant d'electrode
    pOPA2OUT=1;
    dirOPA2OUT=0;
    flag_inversion=0;
    flag_signal_actif=0;
    flag_start=0;

    // attente du demarrage: on doit d'abord passer le potar au maxi puis au mini
    while(1) {
        Delai_ms(500);
        pLED=!pLED; // clignote
        val_potar=ADRESH;
        if( val_potar>240) flag_start=1;
        if( (val_potar<=10) && (flag_start==1)) {
            // start de la production
            pLED=1;
            goto RUNNING;
        }
        if( pSTRAP==0) goto STARTUP;
    }


  RUNNING:
    cpt_minutes=0;
    cpt_secondes=0;
    flag_minutes=0;
    DAC1CON1=MAX_SIGNAL;
    OPA1CON= 0b11010010; // sur DAC
    dirOPA1OUT=1;
    OPA2CON= 0; // OPA2 inhibé
    pOPA2OUT=1;// autre électrode à 20V
    dirOPA2OUT=0;
    flag_sens_signal=1; // sens montant
    flag_signal_actif=1;
    cpt_hache=0;
    TMR6IF=0;
    TMR6IE=1;

    // coupure du moteur: sera active toutes les 4 minutes pendant 10 secondes
    pMOTOR=0;
```

```
    // ici le potar donnera la duree maxi
    // 255= maxi 4 jours 22*255= 5610 minutes
    Delai_ms(500); // attente 500ms pour pouvoir tourner le potar
    Beep();
    Delai_ms(200);
    Beep();
    while(1) {
        pLED=1; // Led fixe
        Delai_ms(500);
        val_potar=ADRESH;
        if( val_potar<=10) {
            // forcage arret
            Beep();Delai_ms(100);Beep();Delai_ms(100);Beep();
            goto STARTUP;
        }
        if( pSTRAP==0) goto STARTUP;
        duree_max=(val_potar-10)*22; // duree max reglage en continu

        if( cpt_minutes>= duree_max) {
                // arrete le signal et attente infinie
                // coupure du moteur pour decantage
                pMOTOR=0;
                OPA1CON= 0; // OPA1 inhibé
                OPA2CON= 0; // OPA2 inhibé
                pOPA2OUT=1;//  électrodes à 24V
                dirOPA2OUT=0;
                pOPA1OUT=1;
                dirOPA1OUT=0;
                flag_signal_actif=0;
                while(1) {
                    Delai_ms(500);
                    pLED= 1;
                    Delai_ms(10);
                    pLED= 0;
                }
        }
        if( flag_minutes==1) {
            flag_minutes=0;
            // ici l'IT a fait evoluer cpt_minutes: on peut tester
           if( ((cpt_minutes & 0x3) ==0) && (cpt_minutes!=0)) {
            // inversion de sens toutes les 4 minutes
            // fait une fois en debut de minute
             if( flag_inversion==1) {
                flag_inversion=0;
                OPA1CON= 0b11010010; // sur DAC
                OPA2CON= 0; // OPA2 inhibé
                pOPA2OUT=1;// autre électrode à 24V
                dirOPA2OUT=0;
             } else {
                flag_inversion=1;
                OPA2CON= 0b11010010; // sur DAC
                OPA1CON= 0; // OPA1 inhibé
                pOPA1OUT=1;// autre électrode à 24V
                dirOPA1OUT=0;
             }

          }
        }
        // activation du moteur pendant 5 secondes toutes les  minutes
        if( cpt_secondes<5) {
            pMOTOR=1;
        } else {
            pMOTOR=0;
        }
    }

    // calibration du potar
    // signal sur OPA1OUT  chaque seconde  de duree en 0.1ms <=> minute reglee
  CALIBRE:
    flag_signal_actif=0; // arrete le signal via IT TMR6
    TMR6IE=1;
    Beep();Delai_ms(100);Beep();Delai_ms(100);
    OPA1CON= 0b11010010; // sur DAC
    dirOPA1OUT=1;
    OPA2CON= 0; // OPA2 inhibé
    pOPA2OUT=1;// autre électrode à 24V
    dirOPA2OUT=0;
    DAC1CON1= MAX_SIGNAL;
    flag_secondes=0;
    while(1) {
        if( pSTRAP==1) goto STARTUP;
        Delai_ms(50);
        val_potar= ADRESH;
        if( val_potar<=10) val_potar=11;
        if( flag_secondes) {
            flag_secondes=0;
            pLED=1;
            DAC1CON1= OFFSET_SIGNAL;// pulse
            Delai_100us((val_potar-10)*22);
            pLED=0;
            DAC1CON1= MAX_SIGNAL;
            flag_secondes=0; // si dépassement de la seconde
        }

    }
}
```

Fichier .HEX de programmation du PIC16F1709 :

```
:060000000000080315528CC
:100008007E14803120007F08F10092189210121D92
:10001800512812117A1E35287A1C1428FD3015280B
:100028007B0822009900FA1D23280130F00070088F
:10003800FB07FE307B02031C2928FA11292801300E
:10004800FB0233307B02031CFA150130F000700804
:100058002000A80706302802031C3528A801013013
:10006800FA060130A2070030A33D07302302D03042
:1000780003192202031C5128A201A3010130F00038
:100088007008A9077A153C302902031C5128A901D8
:100098000130A4070030A53DFA147108FF007E1056
:1000A800090080315728FA01FB0120308400003014
:1000B80085000D308231B3227E1020008031642803
:1000C80082317B2280318231AB2280318131E7213C
:1000D80080310B178B1782319F2280312000B201AB
:1000E80022008D1720000E1C8429FD302200990063
:1000F8002A009101950121000E1122000E158E157E
:10010800021008E112000B1017A12B2018D280E1C37
:100118007428F430AD000130AE00823164228031A1
:1001280020302200 8C0621001C082000AF002F0878
:10013800AC00F1302C02031CA328B201B20A0B3028
:100148002C0203188B283208013A031D8B28220041
:100158008C162000A401A501A901FA10FD30220087
:100168009900D2302A00910021000E152A0095012D
:1001780022008E1521008E11FA157A162000A8018A
:10018800121121001215220 08D13F4302000AD0049
:100198000130AE00823164228031823129228031DF
:1001A800C8302000AD000030AE00823164228031BA
:1001B800823129228031220 08C16F4302000AD00D3
:1001C8000130AE00823164228031210 01C082000F9
:1001D800AF002F08AC000B302C0203180D29823118
:1001E800292280316430200 0AD000030AE00823119
:1001F80064228031823129228031643020 00AD00B0
:100208000030AE00823164228031823129228 0316F
:1002180074280E1C74282C08F63EF200FF300318D0
:100228000030F3001630F4000030F5008231472228
:10023800803173082000A701A7077208A601A60746
:1002480027082502031D2A2926082402031C4F29F2
:1002580022008D132A009101950122008E1521009C
:100268008E1122000E1521000E117A12F430200092
:10027800AD000130AE00823164228031220 08C163C
:100288000A302000AD000030AE0082316422803197
:1002980022008C123A29FA1C7929FA10033020001E
:1002A8002405AF0000302505B00030082F04031DD9
:1002B800792925082404031979293108013A031DED
:1002C8006F29B101D2302A009100950122008E15C4
:1002D80021008E117929B101B10AD2302A00950086
:1002E800910122000E1521000E110530200029026F
:1002F800031881292200 8D17DF2822008D13DF289B
:100308007A122100121582312922803164302000AE
:10031800AD000030AE00823164228031823129226 2
:1003280080316430200 0AD000030AE00823164229C
:10033800803 1D2302A00910021000E152A00950143
:1003480022008E1521008E11FD30220099007A11AD
:1003580020000E1874283230AD000030AE00823113
:10036800642280312 1001C082000AF002F08AC0057
:100378000B302C020318C4290B30AF002F08AC0037
:100388007A1DAC297A1122008C163230990020008F
:100398002C08F63EF200FF3003180030F300163048
:1003A800F4000030F500823147228031730 8F901EA
:1003B800F9077208F801F80782318E22803122008D
:1003C8008C12FD30AA29210095120B1195111515D3
:1003D800951415148B1235302000980027309B0097
:1003E80005309C00210091107C30280096002000E8
:1003F80092104F3028009700210092 10F930280001
:10040800 9D00200012110630280 09E0021001215C0
:100418003230F2007208FB007A102000A201A3011A
:10042800A901A401A5010D3021009D0030309E00D6
:1004380030309F0023000C1521000C159830220045
:10044800980099018C30970008002000AE011930FF
:100458002E020318080022008E161430F80000300F
:10046800F90082318E22823122008E121430F80077
:100478000030F90082318E22823101302000AD0037
:100488002D08AE072B2AF601F701721C4F2A7408B3
:10049800F6077508F73D0130F435F50D890B502A3C
:1004A8000130F336F20C890B552A73087204031DC8
:1004B800492A7708F301F3077608F201F2070800E2
:1004C80020002E08F301F3072D08F201F2070A3085
:1004D800F4000030F500823147228231730 8F901B7
:1004E800F9077208F801F80782318E220800230004
:1004F8000E158E150C150E102100951324000E14E0
:10050800210 08C1222008C1221000C150E148E1260
:10051800 8D13080079087804031908 00FF302000BB
:1005280097009B3096001110111C982A0130F80290
:10053800 0030F93B8E2A8B132000A201A301A901E8
:10054800A001A101A401A501A8018B170800F030A2
:1005580021009900 1A1FAE2A1A180800B02A640050
:0A0568008001013189 0BB42A003430
:020000040001F9
:04000E0084CFFBDEC2
:00000001FF
```