ESCUELA UNIVERSITARIA DE
INGENIERÍA TÉCNICA INDUSTRIAL
DE ZARAGOZA

PROYECTO FINAL DE CARRERA

ANÁLISIS DE LAS POSIBILIDADES DE SENSORES ELECTRÓNICOS PARA LA INTERPRETACIÓN MUSICAL

ANEXO

ALUMNO: ALBERTO PUEYO GIMENO
ESPECIALIDAD: ELECTRÓNICA INDUSTRIAL
DIRECTOR: CHEMA LÓPEZ

CONVOCATORIA: SEPTIEMBRE 2010

ÍNDICE

1.1 INTRODUCCIÓN

El programa que he utilizado para generar los diversos programas de las aplicaciones musicales ha sido el Freescale Codewarrior for HC08 V5.1, y el lenguaje de programación utilizado ha sido el lenguaje C.

Para la transferencia de programas y la comunicación con el micro, he utilizado la tarjeta DEMO9S08QG8.
Para cada apartado de aplicación musical, he descrito el programa empleado mediante diagramas de flujo y el propio código en C, con comentarios.

1.2. APLICACIONES MUSICALES

1.2.1 Aceleración

Diagrama de flujo:

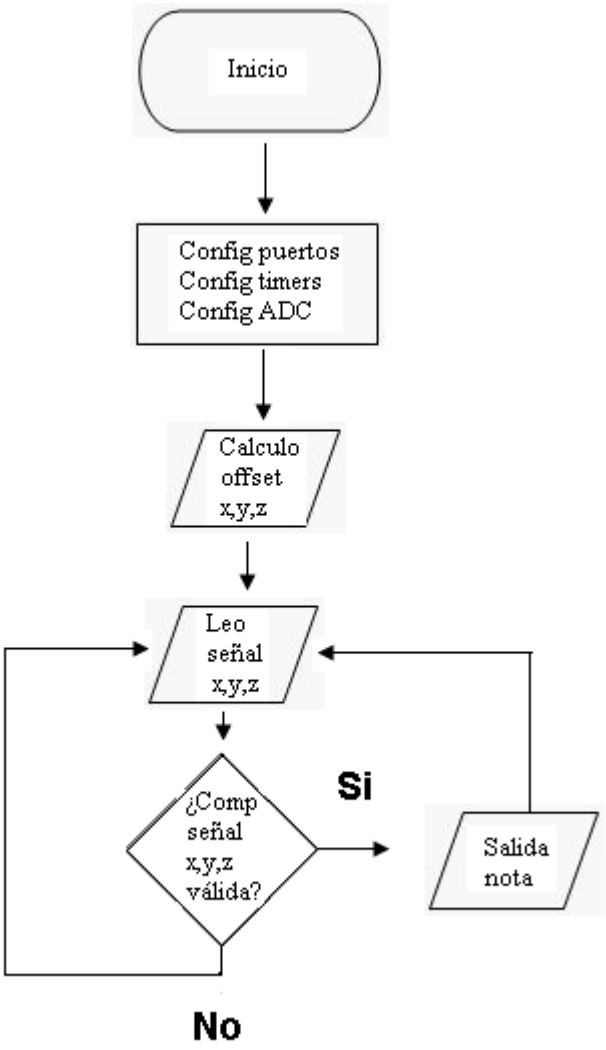
Primero inicializo los puertos A y B y configuro los timers TPM, MTIM y el conversor AD.

Después calculo los offset de cada eje haciendo una media de 32 muestras tomadas.

Entro en el bucle infinito, en el cual, leo y convierto a digital las señales de cada eje y posteriormente efectúo la comparación necesaria para autorizar la nota.

En caso negativo, el ciclo se reinicia y se vuelve al estado de lectura de las señales.

El bucle de tiempo viene determinado por el timer TPM.



Código fuente:

```
lude <hidef.h> /* for EnableInterrupts macro */
#include "derivative.h" /* include peripheral declarations */


 // Declaración de variables globales de este módulo
 unsigned int v,t,x,z,ts=1;              //v,t,x y son variables de control,ts el periodo de muestreo
 unsigned int a1,a2,a3;                  /*Son las pendientes del adsr multiplicadas por 2000*/
 unsigned char salida;                   //salida de la muestra del adsr
  unsigned char ejex=0,ejex2=0,ejey=0,ejey2=0,ejez=0,ejez2=0,offsetx=0,offsety=0,offsetz=0;
 unsigned char control=0,control2,onda,s,n,o,k1,k;                    //Variables de control



 //Declaración de funciones en este módulo
 void IniciarPuertos (void);
 void Retardo (unsigned int tiempo);
 void Timer(void);
 int adsr (unsigned char y1,unsigned char y2,unsigned int t1,unsigned int t2,unsigned int t3,unsigned int
t4);


//Programa principal
void main(void) {


 SOPT1_COPE=0;
IniciarPuertos();
 Timer();
 ADCCFG=0x00;                        //Configuración común del ADC
 ADCSC2=0b00000000;
 APCTL1=0b00000111;
 TPMSC=0b01001110;                    //Configuro el TPM
 TPMMODL=0xFF;
 TPMMODH=0x00;

 s=1;                                //Variable para empezar barrido del ADC

EnableInterrupts; /* enable interrupts */
 /* include your code here */



 for(;;) {                           //Casos con diferentes notas
  switch(n){


   case(1):
  adsr(7,4,200,200,50,50);
  Retardo(ts);
  MTIMCLK=0x03;
  break;


   case(2):
  adsr(5,4,50,50,50,200) ;
  Retardo(ts);
  MTIMCLK=0x04;
  break;


   case(3):
  adsr(7,4,50,50,50,50) ;
  Retardo(ts);
  MTIMCLK=0x06;
```

```
  break;

   case(4):
  adsr(10,4,50,50,50,50) ;
  Retardo(ts);
  MTIMCLK=0x07;

  break;
  }
                        //




   __RESET_WATCHDOG(); /* feeds the dog */
  } /* loop forever */
  /* please make sure that you never leave main */
}



  void IniciarPuertos (void){
  PTADD = 0;              //Puerto A, entradas
  PTBDD = 0xFF;           //Puerto B, salidas
  PTAPE = 0x0F;           //Puerto A,pull-ups
  PTBD = 0x00;            //Puerto B,valor inicial 0
}

  void Retardo (unsigned int tiempo) {        //tiempo en unidades de 1ms
   unsigned char i;

  for (i=0;i<tiempo;i++){
    asm {

      PSHH
      PSHX

      LDHX #400           //constante para 1ms(es 500,pero pongo menos para compensar las operaciones
que se hacen)
bucle: AIX #-1
      CPHX #0
      BNE bucle

      PULX
      PULH
    }
  }
}

void Timer(void){                  //Configuro un timer para generar una onda cuadrada de salida
  MTIMSC=0x40;                     //Configurado para generar interrupcion
 MTIMCLK=0x03;                     //Frecuencia del reloj del bus/256
 MTIMMOD=0x00;                     //Valor para comparar con el contador
}
```

```
/*El vector de interrupción del Timer es el número 12 según
el datasheet*/

interrupt 12 void interrupcionTimer(void){

 if (onda==1){

 onda=0;
 PTBD=salida;
 }
 else {

 onda=1;
    PTBD=0;
 }

  MTIMSC_TRST=1;



}



int adsr(unsigned char y1,unsigned char y2,unsigned int t1,unsigned int t2,unsigned int t3,unsigned int
t4){

if(control==0){

  a1=(y1<<11)/(t1);                  //La constante 2048(desplazar 11 sitios los bits) la utilizo para trabajar
con enteros,y no con reales
a2=((y1-y2)<<11)/(t2);                  //y1 e y2 son las alturas del adsr
a3=(y2<<11)/(t4);
}

 if((salida<y1)&(control==0)){          //Si caigo en la primera rampa

  control2=1;
  salida=(a1*t)>>11;
 t++ ;
 return salida;

 }

 if  (salida>y2) {                  //Si caigo en la segunda rampa(decaimiento)

   control=1;                       //Seteo esta variable para que no caiga en el primer if

   salida=y1-((a2*v)>>11);
   v++;
   return salida;
 }
 if (x<t3){                         //Si caigo en el sostenimiento
  control=1;
  salida=y2;
  x++;
  return salida;
 }
  else{                              //Si caigo en la tercera rampa
```

```
control=1;
salida=y2-((a3*z)>>11);
z++;
if(salida<=0){                    //Si llego a 0 entonces
salida=0;                         //Pongo la salida a 0
control=0;
control2=0;                       //Reseteo las var de control
t=0;
v=0;
x=0 ;
z=0 ;
n=0;
o=0;
ejex=ejex2=255;
ejey=ejey2=255;
ejez=ejez2=255;



 TPMSC=0b01001110;
}
return salida;
}

}
 /*El vector de interrupción del TPM es el número 7 según
el datasheet*/
interrupt 7 void InterrupcionTPM(void){
 if(s==1){
 ADCSC1=0b01000000;                //Configuro entrada patilla PTDA0
 TPMSC_TOF=0;          //Doy ACK al TPM



}
}



 /*El vector de interrupción del ADC es el número 19 según
el datasheet*/
 interrupt 19 void InterrupcionAD(void){


 switch(s){

  case(1):
     offsetx=offsetx+ADCRL;
  k++;
  if(k==32){
  s=2;
  k=0;
```

```
offsetx=offsetx>>5;
   ADCSC1=0b01000001;
}
 ADCSC1=0b01000000;
 break;
```

```
case(2):
  offsety=offsety+ADCRL;
k++;
if(k==32){
s=3;
k=0;
offsety=offsety>>5;
ADCSC1=0b01000001;
  break;
}
 ADCSC1=0b01000010;
 break;

case(3):
  offsetz=offsetz+ADCRL;
k++;
if(k==32){
```

```
s=4;
k=0;
offsetz=offsetz>>5;
  ADCSC1=0b01000000;
  break;
}
 ADCSC1=0b01000010;
 break;
```

```
case(4):
 s=5;
 ejex2=ejex;
ejex=ADCRL;
break;

case(5):
   s=6;
 ejey2=ejey;
ejey=ADCRL;
break;

case(6):
    s=4;
 ejez2=ejez;
ejez=ADCRL;
```

```
 if((ejex>=(ejex2+8))&(ejex>(offsetx+25))){
n=4;
TPMSC=0x00;
break;
}

if((ejey>=(ejey2+8))&(ejey>(offsety+25))){
n=2;
 TPMSC=0x00;
 break;
}
if((ejez>=(ejez2+8))&(ejez>(offsetz+25))){
n=3;
TPMSC=0x00;
break;
}
if((offsetx<100)||(offsety<100)){
o=1;
break;
}
if((o==1)&((offsetx>=140)||(offsety>=140))){
n=1;
TPMSC=0x00;
break;
}
 break;


}
}
```

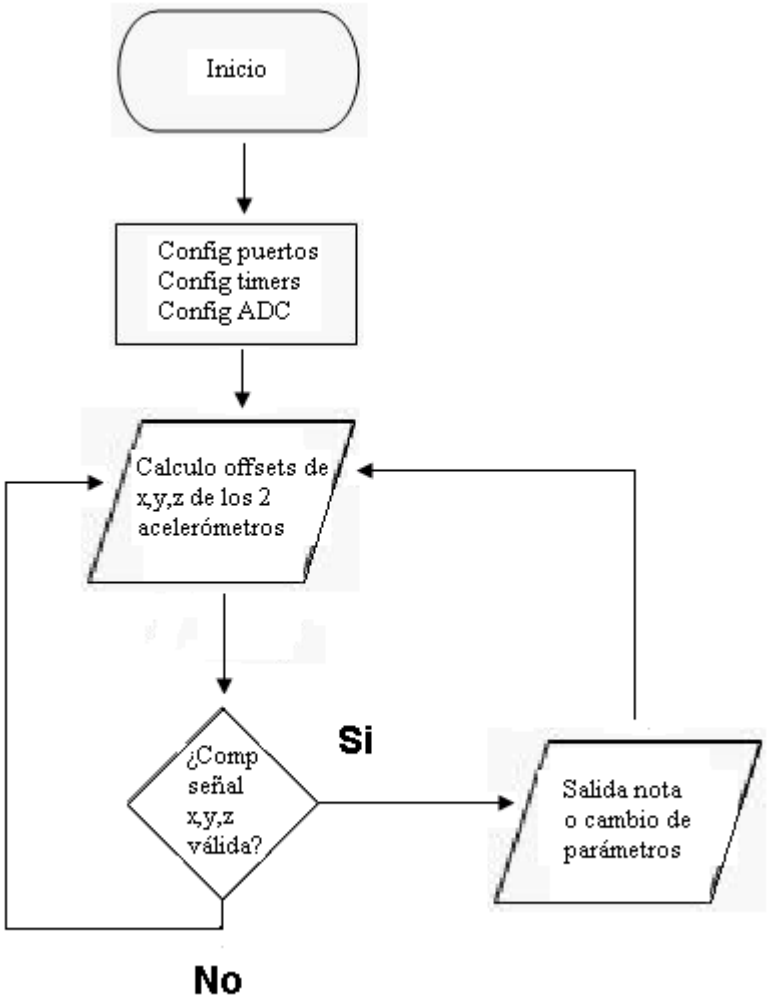
1.2.2 Inclinación

Diagrama de flujo:

Primero inicializo los puertos A y B y configuro los timers TPM , MTIM así como el conversor AD.

Realizo un barrido de conversión de las señales de los ejes x,y,z de cada acelerómetro, de este modo obtengo su tensión de offset en cada posición.

Ejecuto las comparaciones condicionales de cada valor de cada eje, en caso de darse, o se obtiene una nota de salida o se cambian parámetros de tono, volumen o intervalos del ADSR.
En caso negativo, se espera a la interrupción del TPM para volver a calcular los valores de offset.



Código fuente:

```
#include <hidef.h> /* for EnableInterrupts macro */
#include "derivative.h" /* include peripheral declarations */

 // Declaración de variables globales de este módulo
  unsigned char s,k,countx,count,nota,d;
   unsigned int v,t,x,z,ts=1;              //v,t,x y son variables de control,ts el periodo de muestreo
 unsigned int a1,a2,a3;                  /*Son las pendientes del adsr multiplicadas por 2000*/
 unsigned char salida;                   //salida de la muestra del adsr
  signed int offsetx[3];
  signed int offsety[3];
  signed int offsetz[3];

  unsigned char o,o1,y1,y2;          //Variables de control
  unsigned int t1,t2,t3,t4;          //Intervalos del ADSR

    unsigned char control=0,control2,onda,s,n,k1,k2,p;
 //Declaración de funciones en este módulo
 void IniciarPuertos (void);
 void Retardo (unsigned int tiempo);

 void Timer(void);
  int adsr ( unsigned char y1,unsigned char y2,unsigned int t1,unsigned int t2,unsigned int t3,unsigned
int t4);



void main(void) {

  SOPT1_COPE=0;
  ADCCFG=0x00;                    //Configuración común del ADC
  ADCSC2=0b00000000;
  APCTL1=0b00000001;
  TPMSC=0b01001110;                 //Configuro el TPM
  TPMMODL=0xFF;
  TPMMODH=0x00;
  IniciarPuertos();

  Timer();

  EnableInterrupts; /* enable interrupts */
  /* include your code here */

  for(;;) {
   switch(n){

   case(1):                          //Diferentes tipos
  adsr(7,5,200,200,50,200);                //de notas
  Retardo(ts);
  MTIMCLK=0x03;
  break;

   case(2):
  adsr(6,4,50,50,50,50);
  Retardo(ts);
  MTIMCLK=0x03;
  break;
```

```
 case(3):
adsr(4,2,50,50,50,200) ;
Retardo(ts);
MTIMCLK=0x07;
break;

 case(4):
adsr(4,3,50,50,50,200) ;
Retardo(ts);
MTIMCLK=0x07;

  case(5):
   adsr(4,2,200,200,200,200) ;
Retardo(ts);
MTIMCLK=0x05;
break;
   case(6):
   adsr(4,2,200,200,200,200) ;
Retardo(ts);
MTIMCLK=0x05;
break;

case(7):
   adsr(4,2,200,200,200,200) ;
Retardo(ts);

break;
```

```
  case(8):
     adsr(4,2,200,200,200,200) ;
  Retardo(ts);

  break;

   }

   __RESET_WATCHDOG(); /* feeds the dog */
  } /* loop forever */
  /* please make sure that you never leave main */
}


 void IniciarPuertos (void){
  PTADD = 0b00001110;            //Puerto A, entradas
  PTBDD = 0xFF;          //Puerto B, salidas
  PTAPE = 0x01;          //Puerto A,pull-ups
  PTBD = 0x00;           //Puerto B,valor inicial 0
}

 void Retardo (unsigned int tiempo) {       //tiempo en unidades de 1ms

  unsigned char i;

  for (i=0;i<tiempo;i++){
    asm {
```

```
    PSHH
    PSHX

    LDHX #400          //constante para 1ms(es 500,pero pongo menos para compensar las operaciones
que se hacen)
bucle: AIX #-1
    CPHX #0
    BNE bucle

    PULX
    PULH
   }
  }
}

 /*El vector de interrupción del ADC es el número 19 según
 el datasheet*/
  interrupt 19 void InterrupcionAD(void){

 switch(s){

   case(1):

   offsetx[0]=offsetx[0]+ADCRL;     //Calculo el offset del ejex
   k++;
   if(k==8){

   k=0;
offsetx[1]=offsetx[0]>>3;
  s=2;
     offsetx[0]=0;
     PTAD=0b00000010;
  ADCSC1=0b01000000;
  break;
}
 ADCSC1=0b01000000;
 break;

case(2):

 offsety[0]=offsety[0]+ADCRL;     //Calculo el offset del ejey
k++;
if(k==8){

k=0;
offsety[1]=offsety[0]>>3;
  s=3;
     offsety[0]=0;
          PTAD=0b00000100;
  ADCSC1=0b01000000;
  break;
}
 ADCSC1=0b01000000;
 break;

case(3):
```

```
 offsetz[0]=offsetz[0]+ADCRL;      //Calculo el offset del ejez
k++;
if(k==8){

k=0;
offsetz[1]=offsetz[0]>>3;
  s=4;
    offsetz[0]=0;
          PTAD=0b00000110;
  ADCSC1=0b01000000;
  break;
}
 ADCSC1=0b01000000;
 break;

case(4):

offsetx[0]=offsetx[0]+ADCRL;    //Calculo el offset del ejex segundo
                        //acelerometro
k++;
if(k==8){

k=0;
offsetx[2]=offsetx[0]>>3;
  s=5;
    offsetx[0]=0;
          PTAD=0b00001000;
  ADCSC1=0b01000000;
  break;
}
 ADCSC1=0b01000000;
 break;

case(5):

  offsety[0]=offsety[0]+ADCRL;       //Offset del ejey del segundo
k++;                            //acelerómetro
if(k==8){

k=0;
offsety[2]=offsety[0]>>3;
  s=6;
    offsety[0]=0;
        PTAD=0b00001010;
  ADCSC1=0b01000000;
  break;
}
 ADCSC1=0b01000000;
 break;
case(6):
 offsetz[0]=offsetz[0]+ADCRL;       //Offset del ejez del segundo
k++;                            //acelerómetro
if(k==8){

k=0;
```

```
 offsetz[2]=offsetz[0]>>3;
  s=1;
  offsetz[0]=0;
        PTAD=0b00000000;

 if((offsetx[1]<=125)&(offsetx[1]>=115)){    //Comparaciones de inclinación
                                  //para autorizar nota
k1=0;

}
if((offsety[1]<=125)&(offsety[1]>=115)){

k2=0;
}
if(offsetx[2]>=135){
p=1;
}



if(offsetx[2]<=105){
p=2;
}
 if(offsety[2]<=105){
p=3;
}
if(offsety[2]>=135){
p=4;
}
```

```
if((offsetx[1]>=135)&(k1==0)){
switch(p){

 case (1):
 n=1;
 k1=1;
 TPMSC=0x00;
 break;
 case(2):
 n=3;
 k1=1;
TPMSC=0x00;
break;
  case(3):
 n=5;
 k1=1;
TPMSC=0x00;
break;

case(4):
 n=7;
 k1=1;
TPMSC=0x00;
break;
}

}
```

```
if((offsetx[1]<=105)&(k1==0)){
 switch(p){

 case (1):
 n=1;
 k1=1;
 TPMSC=0x00;
 break;
   case(2):
 n=3;
 k1=1;
TPMSC=0x00;
break;
     case(3):
 n=5;
 k1=1;
TPMSC=0x00;
break;

case(4):
 n=7;
 k1=1;
TPMSC=0x00;
break;

 }
 }
if((offsety[1]>=135)&(k2==0)){
 switch(p){

 case (1):
 n=2;
 k2=1;
 TPMSC=0x00;
 break;
    case(2):
 n=4;
 k2=1;
TPMSC=0x00;
break;
    case(3):
 n=6;
 k2=1;
TPMSC=0x00;
break;
case(4):
 n=8;
 k2=1;
TPMSC=0x00;
break;
 }
}
 if((offsety[1]<=105)&(k2==0)){
   switch(p){

 case (1):
```

```
 n=2;
 k2=1;
 TPMSC=0x00;
 break;
    case(2):
 n=4;
 k2=1;
TPMSC=0x00;
break;
    case(3):
 n=6;
 k2=1;
TPMSC=0x00;
break;
 case(4):
 n=8;
 k2=1;
TPMSC=0x00;
break;
}

}

  break;
 }

 ADCSC1=0b01000000;
 break;
  }
  }

   /*El vector de interrupción del TPM es el número 7 según
el datasheet*/
 interrupt 7 void InterrupcionTPM(void){

  s=1;
  PTAD=0x00;
  ADCSC1=0b01000000;                  //Configuro entrada patilla PTDA0
  TPMSC_TOF=0;           //Doy ACK al TPM

 }

  int adsr(unsigned char y1,unsigned char y2,unsigned int t1,unsigned int t2,unsigned int t3,unsigned
int t4){

if(control2==0){

  a1=(y1<<11)/(t1);                  //La constante 2048(desplazar 11 sitios los bits) la utilizo para trabajar
con enteros,y no con reales
a2=((y1-y2)<<11)/(t2);                  //y1 e y2 son las alturas del adsr
a3=(y2<<11)/(t4);

if(n==7){
MTIMCLK=0x03;
}
if(n==8){
```

```
MTIMCLK=0x07;
}
}

 if((salida<y1)&(control==0)){          //Si caigo en la primera rampa

  control2=1;
  salida=(a1*t)>>11;
 t++ ;
 return salida;

 }

 if  (salida>y2) {                    //Si caigo en la segunda rampa(decaimiento)

   control=1;                         //Seteo esta variable para que no caiga en el primer if

   salida=y1-((a2*v)>>11);
   v++;
   if(n==7){
MTIMCLK=0x04;
}
if(n==8){
MTIMCLK=0x06;
}
   return salida;
 }
 if (x<t3){                           //Si caigo en el sostenimiento
  control=1;
  salida=y2;
  x++;
  if(n==7){
MTIMCLK=0x05;
}
 if(n==8){
MTIMCLK=0x05;
}
  return salida;
 }
  else{                               //Si caigo en la tercera rampa
  control=1;
  salida=y2-((a3*z)>>11);
  z++;
   if(n==7){
MTIMCLK=0x06;
}
 if(n==8){
MTIMCLK=0x04;
}
  if(salida<=0){                      //Si llego a 0 entonces
  salida=0;                           //Pongo la salida a 0
  control=0;
  control2=0;                         //Reseteo las var de control
  t=0;
  v=0;
  x=0 ;
```

```
 z=0 ;
 n=0;
 nota=0;
 p=0;
  TPMSC=0b01001110;                  //Habilito de nuevo el TMP
 }
 return salida;
 }

 }
   /*El vector de interrupción del Timer es el número 12 según
el datasheet*/

interrupt 12 void interrupcionTimer(void){
 if (onda==1){
 onda=0;
 PTBD=salida;
 }
 else {

 onda=1;
    PTBD=0;
 }

  MTIMSC_TRST=1;

}
```

```
 void Timer(void){                  //Configuro un timer para generar una onda cuadrada
 MTIMSC=0x40;                       //Configurado para generar interrupcion
MTIMCLK=0x03;                       //Frecuencia del reloj del bus/256
MTIMMOD=0x00;                       //Valor para comparar con el contador
}
```

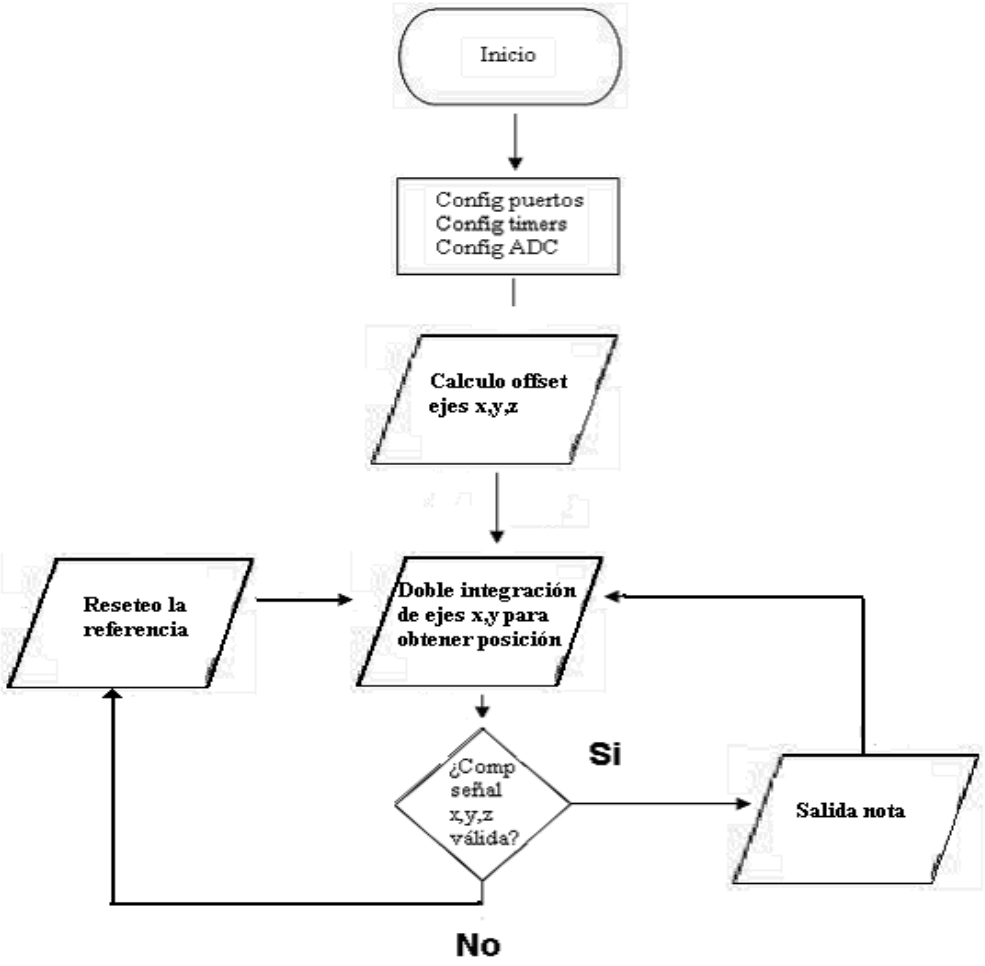
1.2.3 POSICIÓN

Diagrama de flujo

Primero inicializo los puertos A y B y configuro los timers TPM , MTIM así como el conversor AD.

Calculo el valor de offset de cada eje a través de una media de 32 muestras. Después se empieza a ejecutar la rutina cíclica. Cada señal de los ejes **x** e **y** es integrada dos veces a través de una aproximación trapezoidal para obtener la posición en cada momento.

Una vez acabadas las operaciones de integración, se realizan las comparaciones de posición en los ejes **x** e **y** y la aceleración en el **z.**
En caso de que la comparación sea positiva, se reproduce la nota correspondiente, si es negativa, se resetea la posición de referencia y se vuelve al proceso de integración.



Código fuente:

```
#include <hidef.h> /* for EnableInterrupts macro */
#include "derivative.h" /* include peripheral declarations */


 // Declaración de variables globales de este módulo
  unsigned char s,k,countx,count,nota,d;
   unsigned int v,t,x,z,ts=1;              //v,t,x y son variables de control,ts el periodo de muestreo
 unsigned int a1,a2,a3;                  /*Son las pendientes del adsr multiplicadas por 2000*/
 unsigned char salida;                   //salida de la muestra del adsr
  signed int offsetx,offsetz,offsety;      //Offset de cada eje
  signed int acelx[2];                   //Señales de cada eje
  signed int velx[2];
  signed int posx[2];
  signed int acely[2];
  signed int vely[2];
  signed int posy[2];
  signed int acelz[2];
    unsigned char control=0,control2,onda,s,n,o,k1,k; //Variables de control
 //Declaración de funciones en este módulo
 void IniciarPuertos (void);
 void Retardo (unsigned int tiempo);
 void calibrar(void);
 void finalmov(void);
 void Timer(void);
  int adsr (unsigned char y1,unsigned char y2,unsigned int t1,unsigned int t2,unsigned int t3,unsigned
int t4);


void main(void) {


  SOPT1_COPE=0;
  ADCCFG=0x00;                        //Configuración común del ADC
  ADCSC2=0b00000000;
  APCTL1=0b00000111;
  TPMSC=0b01001110;                    //Configuro el TPM
  TPMMODL=0xFF;
  TPMMODH=0x00;
  IniciarPuertos();
  calibrar();
  Timer();
  EnableInterrupts; /* enable interrupts */
  /* include your code here */


  for(;;) {
    switch(n){


    case(1):
   adsr(7,4,200,200,50,50);
   Retardo(ts);
   MTIMCLK=0x03;
   break;


    case(2):
   adsr(5,4,50,50,50,200) ;
   Retardo(ts);
```

```
  MTIMCLK=0x04;
  break;

   case(3):
  adsr(7,4,50,50,50,50) ;
  Retardo(ts);
  MTIMCLK=0x06;
  break;

   case(4):
  adsr(10,4,50,50,50,50) ;
  Retardo(ts);
  MTIMCLK=0x07;

  break;
  }

   __RESET_WATCHDOG(); /* feeds the dog */
 } /* loop forever */
 /* please make sure that you never leave main */
}

 void IniciarPuertos (void){
  PTADD = 0;             //Puerto A, entradas
  PTBDD = 0xFF;          //Puerto B, salidas
  PTAPE = 0x0F;          //Puerto A,pull-ups
  PTBD = 0x00;           //Puerto B,valor inicial 0
}

 void Retardo (unsigned int tiempo) {        //tiempo en unidades de 1ms
   unsigned char i;
  for (i=0;i<tiempo;i++){
    asm {

      PSHH
      PSHX

      LDHX #400          //constante para 1ms(es 500,pero pongo menos para compensar las operaciones
que se hacen)
bucle: AIX #-1
      CPHX #0
      BNE bucle

      PULX
      PULH
    }
  }
}
void calibrar(void){       //Esta función lanza el ADc para calcular los offsets
                    //de cada eje
   s=1;
   ADCSC1=0b01000010;                  //Configuro entrada patilla PTDA0
}
 /*El vector de interrupción del ADC es el número 19 según
 el datasheet*/
   interrupt 19 void InterrupcionAD(void){
```

```
switch(s){

 case(1):

 offsetz=offsetz+ADCRL;       //Cálculo offset eje z
 k++;
 if(k==32){
 s=3;
 k=0;
 offsetz=offsetz>>5;
  ADCSC1=0b01000001;
  break;
 }
  ADCSC1=0b01000010;
  break;

 case(2):                     //Integración señal eje x
 acelx[1]=(ADCRL-offsetx)>>1;
 if ((acelx[1] <=2)&&(acelx[1] >= -2)) //Ventana de discriminación
   {acelx[1] = 0;}

 //Primera integración
  velx[1]= velx[0]+ acelx[0]+ ((acelx[1] -acelx[0])>>1);

  //Segunda integración
   posx[1]= posx[0] + velx[0] + ((velx[1] - velx[0])>>1);

   acelx[0]=acelx[1];
   velx[0]=velx[1];
   posx[0]=posx[1];

   if(((acelx[1]>=10)||(acelx[1]<=-10))&&((posx[1]>=-500)||(posx[1]<=500))){   //Condición para
que al iniciar movimiento no se reseteen los parámetros
       d=0;                                          //cuando acelx==0(velocidad constante)
   }

    s=6;
    ADCSC1=0b01000001;

  break;

 case(3):
  offsety=offsety+ADCRL;       //Cálculo offset eje y
 k++;
 if(k==32){
 s=5;
 k=0;
 offsety=offsety>>5;
  ADCSC1=0b01000000;
   break;
 }
  ADCSC1=0b01000001;
  break;

 case(4):
```

```
  acelz[1]=ADCRL-offsetz;
  s=2;

   if(((posy[1]>=5000)||(posy[1]<=-5000))&&((posx[1]>=5000)||(posx[1]<=-
5000))&&((k1==0)&&(acelz[1]>=17))){
    n=2;
   k1=1;
   d=1;
  TPMSC=0x00;
   }
  if(((posx[1]>=10000)||(posx[1]<=-13000))&&((k1==0)&&(acelz[1]>=17))){
    n=1;
   k1=1;
   d=1;
  TPMSC=0x00;

   }
     if((((posy[1]>=10000)||(posy[1]<=-10000))&&((k1==0)&&(acelz[1]>=17))){
   n=1;
   k1=1;
   d=1;
  TPMSC=0x00;
   }
  if(((posx[1]>=4000)||(posx[1]<=-5000))&&((k1==0)&&(acelz[1]>=17))){
    n=4;
   k1=1;
   d=1;
  TPMSC=0x00;
   }

   if(((posy[1]>=4000)||(posy[1]<=-5000))&&((k1==0)&&(acelz[1]>=17))){
   n=4;
  k1=1;
  d=1;
 TPMSC=0x00;
   }


break;

case(5):

 offsetx=offsetx+ADCRL;        //Cálculo offset eje x
k++;
if(k==32){
s=2;
k=0;
offsetx=offsetx>>5;

}
 ADCSC1=0b01000000;
 break;

case(6):                       //Integración señal eje y
 acely[1]=(ADCRL-offsety)>>1;
if ((acely[1] <=2)&&(acely[1] >= -2)) //Ventana de discriminación
```

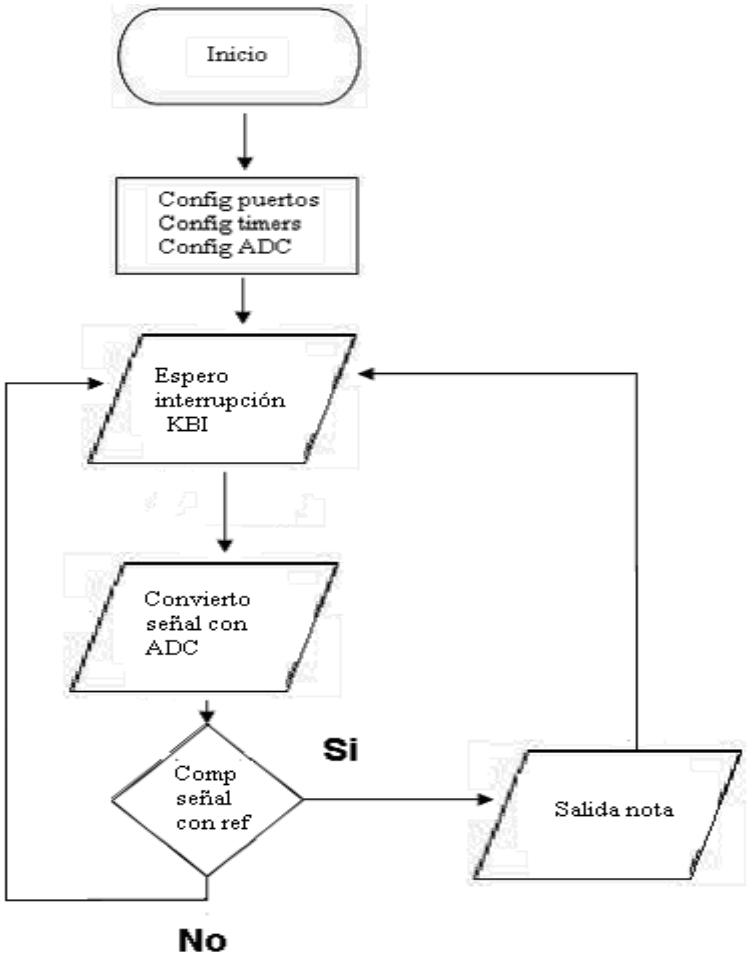
```
    {acely[1] = 0;}

  //Primera integración
   vely[1]= vely[0]+ acely[0]+ ((acely[1] -acely[0])>>1);

   //Segunda integración
    posy[1]= posy[0] + vely[0] + ((vely[1] - vely[0])>>1);

    acely[0]=acely[1];
    vely[0]=vely[1];
    posy[0]=posy[1];

     if(((acely[1]>=10)||(acely[1]<=-10))&&((posy[1]>=-500)||(posy[1]<=500))){   //Condición para
que al iniciar movimiento no se reseteen los parámetros
        d=0;                                          //cuando acelx==0(velocidad constante)
    }

    finalmov();
      s=4;
     ADCSC1=0b01000010;

   break;

 }
  }

  /*El vector de interrupción del TPM es el número 7 según
el datasheet*/
interrupt 7 void InterrupcionTPM(void){

 ADCSC1=0b01000000;                //Configuro entrada patilla PTDA0
 TPMSC_TOF=0;          //Doy ACK al TPM

}

void finalmov(void){      //Esta función se encarga de resetear los parámetros
                  //cuando no hay movimiento
 if ((acelx[1]==0) &&(d==1)&&(acely[1]==0))  //contamos el número de muestras que son 0
  { countx++;}
else { countx =0;}
 if ((acelx[1]==0) &&(d==0)&&(acely[1]==0))  //contamos el número de muestras que son 0
  { count++;}

 if ((count>=200))        //si superamos las 25,asumimos que la velocidad es cero y reseteamos
posición
  {
   posx[1]=0;
  posx[0]=0;
  velx[1]=0;
  velx[0]=0;
  k1=0;
  count=0;
   posy[1]=0;
  posy[0]=0;
  vely[1]=0;
  vely[0]=0;
```

```
  }
  if ((countx>=35))          //si superamos las 25,asumimos que la velocidad es cero y reseteamos posición
  {
  count=0;
  posx[1]=0;
  posx[0]=0;
  velx[1]=0;
  velx[0]=0;
  k1=0;
    posy[1]=0;
  posy[0]=0;
  vely[1]=0;
  vely[0]=0;

  }

 }

  int adsr(unsigned char y1,unsigned char y2,unsigned int t1,unsigned int t2,unsigned int t3,unsigned int t4){
if(control==0);
  a1=(y1<<11)/(t1);                  //La constante 2048(desplazar 11 sitios los bits) la utilizo para trabajar con enteros,y no con reales
a2=((y1-y2)<<11)/(t2);                  //y1 e y2 son las alturas del adsr
a3=(y2<<11)/(t4);
}
if((salida<y1)&(control==0)){          //Si caigo en la primera rampa

 control2=1;
 salida=(a1*t)>>11;
t++ ;
return salida;
}
if  (salida>y2) {                    //Si caigo en la segunda rampa(decaimiento)
  control=1;                         //Seteo esta variable para que no caiga en el primer if
  salida=y1-((a2*v)>>11);
  v++;
  return salida;
}
if (x<t3){                           //Si caigo en el sostenimiento
 control=1;
 salida=y2;
 x++;
 return salida;
}
 else{                                //Si caigo en la tercera rampa
 control=1;
 salida=y2-((a3*z)>>11);
 z++;
 if(salida<=0){                       //Si llego a 0 entonces
 salida=0;                           //Pongo la salida a 0
 control=0;
 control2=0;                          //Reseteo las var de control
 t=0;
```

```
 v=0;
 x=0 ;
 z=0 ;
 n=0;
 nota=0;

  TPMSC=0b01001110;
 }
 return salida;
 }

 }
   /*El vector de interrupción del Timer es el número 12 según
el datasheet*/

interrupt 12 void interrupcionTimer(void){

 if (onda==1){

 onda=0;
 PTBD=salida;
 }
 else {
 onda=1;
    PTBD=0;
 }

  MTIMSC_TRST=1;
 }
   void Timer(void){              //Configuro un timer para generar una onda cuadrada
  MTIMSC=0x40;                    //Configurado para generar interrupcion
 MTIMCLK=0x03;                    //Frecuencia del reloj del bus/256
 MTIMMOD=0x00;                    //Valor para comparar con el contador
}
```

1.2.4 PELOTA MUSICAL

Diagrama de flujo



Primero configuro los puertos, las interrupciones del timer MTIM y del KBI, así como el conversor AD. Espero la interrupción del KBI por la patilla 0 del puerto A. Después convierto el pulso de la interrupción a analógico con el ADC y lo comparo con un nivel, para comprobar que no lo ha disparado el posible ruido. Si la comparación es positiva, se reproduce una nota al azar entre 6 tipos posibles, si no, se vuelve al estado de espera del KBI.

Código fuente:

```
#include <hidef.h> /* for EnableInterrupts macro */
#include "derivative.h" /* include peripheral declarations */

 // Declaración de variables globales de este módulo
  unsigned char s,k,countx,count,nota,d;
   unsigned int v,t,x,z,ts=1;              //v,t,x y son variables de control,ts el periodo de muestreo
 unsigned int a1,a2,a3;                 /*Son las pendientes del adsr multiplicadas por 2000*/
 unsigned char salida;                  //salida de la muestra del adsr
                                 //variables de control

  unsigned char o,o1,y1,y2;
  unsigned int t1,t2,t3,t4;
  signed int acelz[2];
    unsigned char control=0,control2,onda,s,n,k1,k2,p,k,r,l,b,d1,j,w; //Variables de control
 //Declaración de funciones en este módulo
 void IniciarPuertos (void);
 void Retardo (unsigned int tiempo);

 void Timer(void);
  int adsr ( unsigned char y1,unsigned char y2,unsigned int t1,unsigned int t2,unsigned int t3,unsigned int t4);

void main(void) {

  SOPT1_COPE=0;
      IniciarPuertos();
 KBISC=0b00000110;
 KBIPE=0b00000001;
 KBIES=0b00000001;
 ADCCFG=0x00;

                        //Configuración común del ADC
 ADCSC2=0b00000000;

 Timer();

  EnableInterrupts; /* enable interrupts */
  /* include your code here */

  for(;;) {              //Bucle infinito, cada caso es un tipo de nota

   if(w==1){    //Variable de autorización de nota

    switch(n){
```

```
 case(1):
adsr(10,6,200,200,200,200);
Retardo(ts);
MTIMCLK=0x03;
break;
 case(2):
adsr(10,6,50,50,50,50);
Retardo(ts);
MTIMCLK=0x04;
break;
 case(3):
adsr(10,6,200,200,200,200) ;
Retardo(ts);
MTIMCLK=0x05;
break;

 case(4):
adsr(10,6,50,50,50,50) ;
Retardo(ts);
MTIMCLK=0x06;

   case(5):
    adsr(10,6,200,200,200,200) ;
Retardo(ts);
   break;
      case(6):
      adsr(10,6,50,50,50,50) ;
   Retardo(ts);
   MTIMCLK=0x03;
   break;

    }
   } else{       //Con este contador se ejecuta una nota aleatoriamente
   n++;
   if(n==7){
   n=1;
   }
   }
    __RESET_WATCHDOG(); /* feeds the dog */
  } /* loop forever */
  /* please make sure that you never leave main */
}

 void IniciarPuertos (void){
  PTADD = 0b00000000;            //Puerto A, entradas
  PTBDD = 0xFF;          //Puerto B, salidas
  PTAPE = 0xF0;          //Puerto A,pull-ups
```

```
 PTBD = 0x00;            //Puerto B,valor inicial 0
}
 void Retardo (unsigned int tiempo) {        //tiempo en unidades de 1ms
   unsigned char i;
  for (i=0;i<tiempo;i++){
    asm {

      PSHH
      PSHX

      LDHX #400          //constante para 1ms(es 500,pero pongo menos para
compensar las operaciones que se hacen)
bucle: AIX #-1
      CPHX #0
      BNE bucle

      PULX
      PULH
    }
  }
}

   /*El vector de interrupción del KBI es el número 18 según
 el datasheet*/
 interrupt 18 void InterrupcionKBI(void){
  KBIPE_KBIPE0=0;
  KBISC_KBIE=0;
  APCTL1=0b00000001;  //Configuro el ADC y lanzo la conversión

ADCSC1=0b01000000;

KBISC_KBACK=1;
 }

/*El vector de interrupción del ADC es el número 19 según
 el datasheet*/
  interrupt 19 void InterrupcionAD(void){


  if(ADCRL>=235){                    //Comparo el valor de la conversión para
autorizar la nota                          //y ver que no ha sido un pulso de
ruido
  w=1;



  APCTL1=0b00000000;
  } else  {
```

```
        APCTL1=0b00000000;          //Desactivo el ADC y activo el KBI para
volver a empezar
        KBISC_KBIE=1;
    KBIPE_KBIPE0=1;


   }


   }
```

```
   int adsr(unsigned char y1,unsigned char y2,unsigned int t1,unsigned int
t2,unsigned int t3,unsigned int t4){


if(control2==0){


  a1=(y1<<11)/(t1);                  //La constante 2048(desplazar 11 sitios los bits)
la utilizo para trabajar con enteros,y no con reales
a2=((y1-y2)<<11)/(t2);                  //y1 e y2 son las alturas del adsr
a3=(y2<<11)/(t4);
 if(n==5){
MTIMCLK=0x03;
}
}
 if((salida<y1)&(control==0)){          //Si caigo en la primera rampa


  control2=1;
  salida=(a1*t)>>11;
 t++ ;
 return salida;
 }
 if  (salida>y2) {                  //Si caigo en la segunda rampa(decaimiento)


   control=1;                        //Seteo esta variable para que no caiga en el primer
if


   salida=y1-((a2*v)>>11);
   v++;
      if(n==5){
MTIMCLK=0x04;
}
   return salida;
 }
```

```
 if (x<t3){                    //Si caigo en el sostenimiento
  control=1;
  salida=y2;
  x++;
   if(n==5){
MTIMCLK=0x05;
}

  return salida;
 }
  else{                         //Si caigo en la tercera rampa
  control=1;
  salida=y2-((a3*z)>>11);
  z++;
   if(n==5){
MTIMCLK=0x06;
}
  if(salida<=0){                //Si llego a 0 entonces
  salida=0;                     //Pongo la salida a 0
  control=0;
  control2=0;                   //Reseteo las var de control
  t=0;
  v=0;
  x=0 ;
  z=0 ;
  w=0;
  p=0;
   KBIPE_KBIPE0=1;
  KBISC_KBIE=1;


  }
  return salida;
  }

  }
   /*El vector de interrupción del Timer es el número 12 según
el datasheet*/

interrupt 12 void interrupcionTimer(void){

 if (onda==1){

 onda=0;
 PTBD=salida;
 }
 else {
```

```
 onda=1;
    PTBD=0;
 }
  MTIMSC_TRST=1;
}
  void Timer(void){                  //Configuro un timer para generar una onda
cuadrada
 MTIMSC=0x40;                        //Configurado para generar interrupcion
MTIMCLK=0x03;                        //Frecuencia del reloj del bus/256
MTIMMOD=0x00;                        //Valor para comparar con el contador
}
```