APENDICE

FIRMWARE

```
#include <18F4550.h>
#device ADC=8
#fuses
HS,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,CPUDIV1,VREGEN//HSPLL,L
VP
#use delay(clock=20000000)
#use rs232(baud=19200, xmit=PIN_C6, rcv=PIN_C7)

#define USB_HID_DEVICE     FALSE
#define USB_EP1_TX_ENABLE  USB_ENABLE_BULK
#define USB_EP1_RX_ENABLE  USB_ENABLE_BULK
#define USB_EP1_TX_SIZE    64
#define USB_EP1_RX_SIZE    8

#include <pic18_usb.h>
#include <PicUSB.h>
#include <usb.c>
#define dato2      recibe[2]
#define dato1      recibe[1]
#define modo       recibe[0]
```

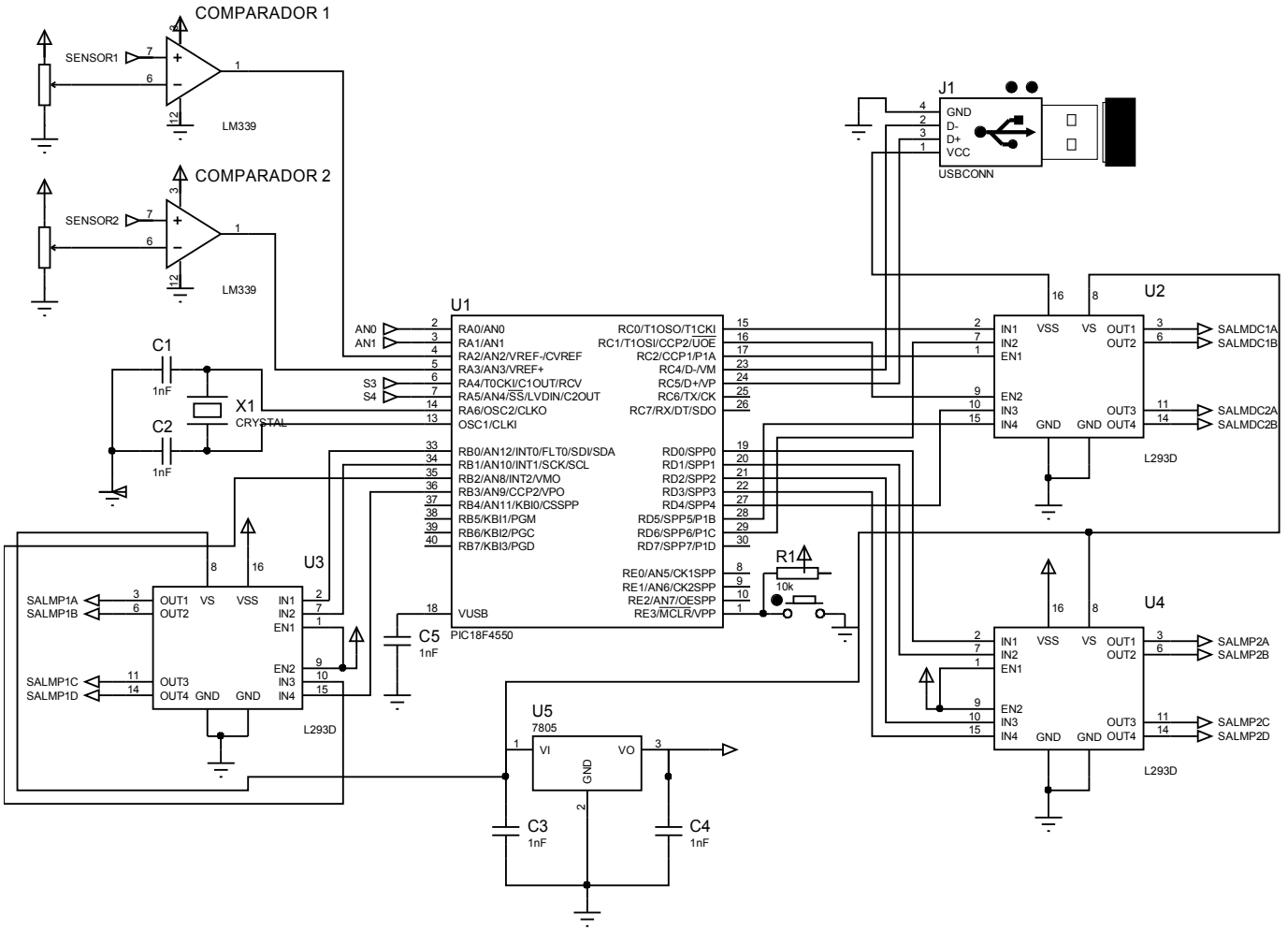
```
#BYTE TRISD=0XF95
#BYTE PORTD=0XF83
#use rtos(timer=1,minor_cycle=500us)

struct PORTD_BITS
  {
  BYTE MP: 4;
  BYTE MDC2 :   2;
  BYTE MDC1: 1;
  BYTE NO:    1;
  };
struct PORTD_BITS portd_bits;

int8 recibe[3];
int8 salida=0x33;
int8 valorADC[1], sensores[1];
int8 opcion, cuenta=0,cuenta2=0;
int1 dirMP, ddMP, dirMDC, ddMDC, MPon=0, MDCon=0,aux=0;
int1 dirMP2, ddMP2, dirMDC2, ddMDC2, MPon2=0, MDCon2=0;
int8  velMP=0,  velMDC;
int32 durMP, durMDC;
int8  velMP2=0,  velMDC2;
int32 durMP2, durMDC2;
int edomotor=1;

BYTE const posicion[4]={0b1100, 0b0110,0b0011,0b1001};
#task (rate=500us,max=500us)
void recibep();
//////////////       motor de pulsos  ///////////////
#task (rate=500us,max=500us)
void giroMP();
//////////////       motor de pulsos 2  //////////////
#task (rate=500us,max=500us)
void giroMP2();
//////////////       motor DC  ///////////////////
#task (rate=1ms,max=500us)
void giroMDC2();
//////////////       motor DC  ///////////////////
#task (rate=1ms,max=500us)
void giroMDC();

//////////////  Lectura entrada analógica  /////////
void senAn(int canal)
{
    set_adc_channel(canal);
```

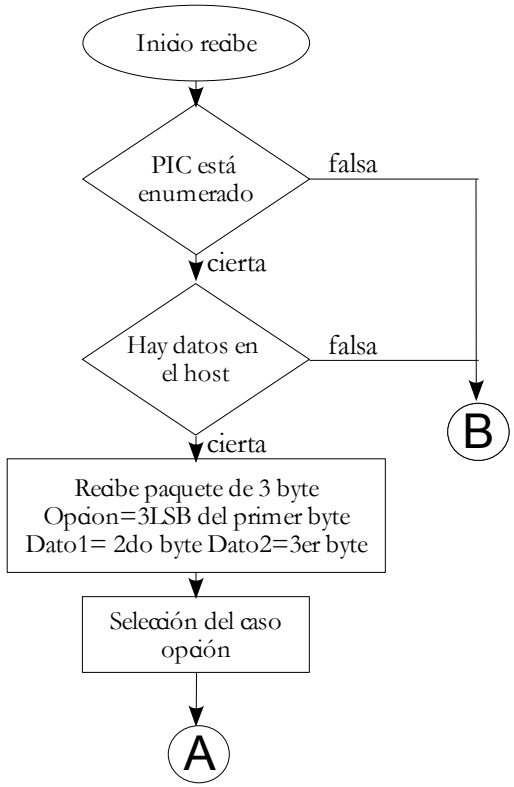
```
    delay_us(1000);
    valorADC[0]=read_adc();
    usb_put_packet(1,valorADC,1,USB_DTS_TOGGLE);
}

///////////////   lectura de los sensores digitales   //
void senDig(void)
{
    sensores[0]=0;
    if (input(PIN_A2)) bit_set(sensores[0],0);
    if (input(PIN_A3)) bit_set(sensores[0],1);
    if (input(PIN_A4)) bit_set(sensores[0],2);
    if (input(PIN_A5)) bit_set(sensores[0],3);

    usb_put_packet(1,sensores,1,USB_DTS_TOGGLE);
}


void main(void)
{
    setup_adc(ADC_CLOCK_INTERNAL);

setup_adc_ports(AN0_TO_AN1_ANALOG);
    TRISD=0X00;
    portd_bits=0x00;
    PORTD = portd_bits;
    output_C(0);
    output_low(PIN_B6);
    output_high(PIN_B7);
    printf("Inicializando\n");
    usb_init();
    printf("Inicializacion terminada\n");
    usb_task();
    printf("Esperando enumeracion\n");
    usb_wait_for_enumeration();
    output_low(PIN_C0);
    output_low(PIN_B7);
    output_high(PIN_B6);
    setup_timer_2(T2_DIV_BY_16, 127, 1);
    rtos_disable(giroMP);
    rtos_disable(giroMP2);
    rtos_disable(giroMDC);
    rtos_disable(giroMDC2);
    rtos_run();
}

void recibep()
{
    rtos_await(usb_enumerated());
        if (usb_kbhit(1))
    {
        usb_get_packet(1,recibe, 3);
        opcion=modo&15;

        switch (opcion)
        case 1:
            if (dato1!=0)
            {
                durMP=(dato1*25);
                velMP=dato2;
                dirMP=bit_test(modo,7);
                ddMP= bit_test(modo,6);
                MPon=1;
                rtos_enable(giroMP);
            }
            else
            {
                MPon=0;
                output_b(0);
                rtos_disable(giroMP);
            }
        break;
        case 2:
            if (dato1!=0)
```

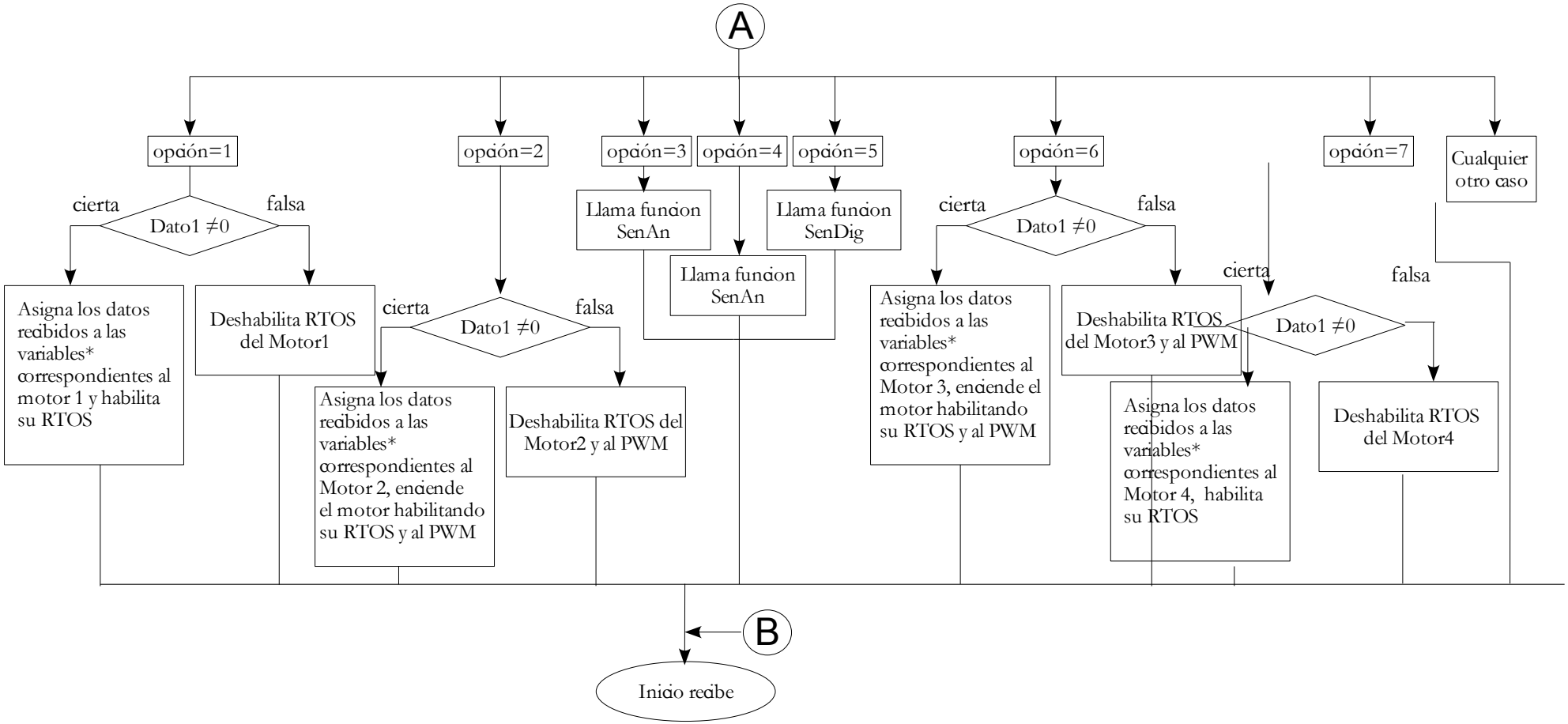
```
    {
        durMDC=dato1*1000;
        velMDC=dato2;
        ddMDC=bit_test(modo,6);
        rtos_enable(giroMDC);
        if (bit_test(modo,7))
         {
            portd_bits.MDC1=0;
            PORTD=portd_bits;
            output_high(PIN_C0);
        }
        else if (!bit_test(modo,7))
        {
            portd_bits.MDC1=1;
            PORTD=portd_bits;
            output_low(PIN_C0);
        }
        setup_ccp1(CCP_PWM);
        set_pwm1_duty(velMDC);
    }
    else
    {
        portd_bits.MDC1=0;
        PORTD=portd_bits;
        output_low(PIN_C0);
        setup_ccp1(CCP_OFF);
        rtos_disable(giroMDC);
    }
break;
case 3:
    senAn(0);
break;
case 4:
    senAn(1);
break;
case 5:
    senDig();
break;
case 6:
    if (dato1!=0)
    {
        durMP2=(dato1*25);
        velMP2=dato2;
        dirMP2=bit_test(modo,7);
        ddMP2= bit_test(modo,6);
        MPon2=1;
        rtos_enable(giroMP2);
    }
    else
    {
        MPon2=0;
        rtos_disable(giroMP2);
    }
break;
case 7:
    if (dato1!=0)
    {
        durMDC2=dato1*1000;
        velMDC2=dato2;
        dirMDC2=bit_test(modo,7);
        ddMDC2=bit_test(modo,6);
        rtos_enable(giroMDC2);
        if (dirMDC2==1)
        {
            portd_bits.MDC2=0x01;
            PORTD=portd_bits;
        }
        else if (dirMDC2==0)
        {
            portd_bits.MDC2=0x02;
            PORTD=portd_bits;
        }
        setup_ccp2(CCP_PWM);
        set_pwm2_duty(velMDC2);
    }
    else
    {
        portd_bits.MDC2=0;
        PORTD=portd_bits;
        setup_ccp2(CCP_OFF);
        MDCon2=0;
        rtos_disable(giroMDC2);
    }
break;
default:
break;
}
rtos_yield();
}
void giroMP()
```

```
{
    rtos_await( Mpon==1);
    ++cuenta;
    if (cuenta>velMP)
    {
        if (dirMP==1)  rotate_left(&salida,1);
        else   rotate_right(&salida,1);
        delay_us(800);
        output_b(salida);
        cuenta=0;
        if (ddMP==1)  --durMP;
        if (durMP==0) rtos_disable(giroMP);
    }
   rtos_yield();
}
void giroMP2()
{
    rtos_await( MPon2==1);
    ++cuenta2;
    if (cuenta2>velMP2)
    {
        if (dirMP2==1)
edomotor=((edomotor+1)&3);
        else edomotor=((edomotor-1)&3);
        delay_us(800);
        portd_bits.MP = posicion[edomotor];
        PORTD=portd_bits;
        cuenta2=0;
        if (ddMP2==1)  --durMP2;
        if (durMP2==0) rtos_disable(giroMP2);
    }
     rtos_yield();
}
void giroMDC2(void)
{
    rtos_await(ddMDC2==1);
    --durMDC2;
    if (durMDC2==0)
    {
        portd_bits.MDC2=0;
        PORTD=portd_bits;
        setup_ccp2(CCP_OFF);
        rtos_disable(giroMDC2);
    }
    rtos_yield();
}
 void giroMDC(void)
{
    rtos_await(ddMDC==1);
    --durMDC;
    if (durMDC==0)
    {
        portd_bits.MDC1=0;
        PORTD=portd_bits;
        output_low(PIN_C0);
        setup_ccp1(CCP_OFF);
        rtos_disable(giroMDC);
    }
    rtos_yield();
}
```

DIAGRAMA DE CONEXIONES




Inicio recibe
PIC está enumerado
falsa
cierta
Hay datos en el host
falsa
B
cierta
Recibe paquete de 3 byte
Opcion=3LSB del primer byte
Dato1= 2do byte Dato2=3er byte
Selección del caso opción
A


A
opción=1
opción=2
opción=3
opción=4
opción=5
opción=6
opción=7
Cualquier otro caso
cierta
falsa
Dato1 ≠0
Asigna los datos recibidos a las variables* correspondientes al motor 1 y habilita su RTOS
Deshabilita RTOS del Motor1
Asigna los datos recibidos a las variables* correspondientes al Motor 2, enciende el motor habilitando su RTOS y al PWM
Deshabilita RTOS del Motor2 y al PWM
Llama funcion SenAn
Llama funcion SenDig
Llama funcion SenAn
Asigna los datos recibidos a las variables* correspondientes al Motor 3, enciende el motor habilitando su RTOS y al PWM
Deshabilita RTOS del Motor3 y al PWM
Asigna los datos recibidos a las variables* correspondientes al Motor 4, habilita su RTOS
Deshabilita RTOS del Motor4
B
Inicio recibe

PRUEBA DE LA INTERFAZ

La prueba de la interfaz se realizó con niños de edad primaria de 6 a 8 años. La actividad estuvo dividida en tres etapas: diseño, armado y operación mediante la interfaz.

La temática propuesta a los niños fue la creación de una feria de diversiones y la aceptaron con agrado. En la etapa de diseño, se les pidió que pensaran en juegos mecánicos que tienen movimiento rotatorio, después que eligieran alguno y lo dibujara en papel.

Fotografía 1. Niños plasmando sus ideas en papel

La siguiente etapa fue construir el juego mediante material de reuso como botellas de plástico, tapas, envases, cajas, estambre, etc.. Se presentó un poco de dificultad sobre todo en los niños más chicos al tener que pasar su diseño de papel a maqueta. Sin embargo fue una gran oportunidad para el desarrollo de la creatividad, ya que ellos eligieron los materiales a utilizar.

Fotografía 2. Construyendo las maquetas

Por último se adaptó cada una de las maquetas a los motores de la interfaz. Se probó con el programa “Control independiente” para conocer como se constituía la interfaz electrónica, después se dejo que los niños trabajaran libremente con el programa “Principios de programación”

Fotografía 3. Probando su maqueta con la interfaz

Gracias a los niños Andrés Daniel, Dulce, Isabel, Jesús, Kevin, Natalia, Sara y Xantil que participaron en la puesta en marcha de la interfaz.