# Apéndice 2
# Código fuente

### a) Funcionamiento de los puertos B y D del microcontrolador

El siguiente código fuente tiene como objetivo mostrar el funcionamiento de los puertos B y D del microcontrolador PIC16F877, de cómo configurar sus entradas y salidas, encendiendo o apagando dos leds si las condiciones se cumplen.

**Lectura y salida de datos.**

```
#include <16F877.h>   // Configuración general del microcontrolador
#include <STDLIB.H>
#fuses HS,NOWDT,NOPROTECT
#use delay(clock=10000000)
#use rs232(BAUD=9600, xmit=PIN_C6, rcv=PIN_C7)

void main() {

  SET_TRIS_B( 0x00 );  // Salidas del Puerto B
  SET_TRIS_D( 0XFF); // Entradas del Puerto D

  if (!input(PIN_D2) && !input (PIN_D3)) // Condiciones Pin D2 y D3 a 0
    {
    printf("Uno");    //Imprimir un uno en pantalla si condición se cumple
    output_low(PIN_B6); // Pin B6 a cero
    output_low(PIN_B7); // Pin B7 a cero
    }
    else
     {
     if (!input(PIN_D2) && input (PIN_D3)) //Condiciones Pin D2 a 0 y D3 a 1
      {
       printf("Dos"); //Imprimir un dos en pantalla si condición se cumple
       output_low(PIN_B6);  // Pin B6 a cero
       output_high(PIN_B7); // Pin B7 a uno
      }
      else
      {
       if (input(PIN_D2) && !input (PIN_D3)) // Condiciones Pin D2 a 1 y D3 a 0
       {
       printf("Tres"); //Imprimir un tres en pantalla si condición se cumple
       output_high(PIN_B6); // Pin B6 a uno
       output_low(PIN_B7);  // Pin B7 a cero
       }
       else
          {  printf("Cuatro"); //Imprimir un cuatro en pantalla si condición se cumple
          output_high(PIN_B6); // Pin B6 a uno
```

```
        output_high(PIN_B7); // Pin B7 a uno }
     }
    }
}
```

**b) Lectura de sensores.**

```
#include <16F877.h> // Configuración general del microcontrolador
#include <STDLIB.H>
#include <math.h>
#fuses HS,NOWDT,NOPROTECT
#use delay(clock=4000000)
#use rs232(BAUD=2400, xmit=PIN_C6, rcv=PIN_C7)

main()
{
 int d,i;  //Configuracion variables
 float valor1,g1,valor2,g2;
 signed long m1,m2,val1,val2;
 setup_adc(ADC_CLOCK_INTERNAL);
 setup_adc_ports(ALL_ANALOG);
 SET_TRIS_B( 0x00 ); // Salidas Puerto B
 SET_TRIS_D( 0XFF); // Entradas Puerto D
 d=0;  // Variable para indicar bandera derecha
 i=0;   // Variable para indicar bandera izquierda
 while(1)
 {
 set_adc_channel(1);  // Canal 1 toma valores para el sensor 1
 delay_ms(20);
 valor1 = read_adc(); // Guardar valores de sensor 1
 valor1=valor1*5/255; // Sacando los grados de inclinación
 g1=(valor1-2.5)/1.2;
 g1=asin(g1);
 g1=g1*(180/3.141596); //Conversión de radianes a grados
 m1=g1;
 delay_ms( 100 );
 set_adc_channel(2); // Canal 2 tomo valores para el sensor 2
 delay_ms(20);
 valor2 = read_adc();// Guardar valores de sensor 2
 valor2=valor2*5/255;
 g2=(valor2-2.5)/1.2;
 g2=asin(g2);
```

```
g2=g2*(180/3.141596); //Conversión de radianes a grados
}
}
```

### c) Transmisión de datos

Programa que se encarga de configurar los elementos necesarios del PIC microcontrolador para recibir, procesar y enviar la información al receptor.

A fin de que se entienda el funcionamiento del siguiente código fuente, se comenta.

```
#include <16F877.h> // Configuración general del microcontrolador
#include <STDLIB.H>
#include <math.h>
#fuses HS,NOWDT,NOPROTECT
#use delay(clock=4000000)
#use rs232(BAUD=2400, xmit=PIN_C6, rcv=PIN_C7)

main()
{
 int d,i;  //Configuracion variables
 float valor1,g1,valor2,g2;
 signed long m1,m2,val1,val2;
 setup_adc(ADC_CLOCK_INTERNAL);
 setup_adc_ports(ALL_ANALOG);
 SET_TRIS_B( 0x00 ); // Salidas Puerto B
 SET_TRIS_D( 0XFF); // Entradas Puerto D
 d=0;  // Variable para indicar bandera derecha
 i=0;   // Variable para indicar bandera izquierda
 while(1)
 {
 set_adc_channel(1);  // Canal 1 toma valores para el sensor 1
 delay_ms(20);
 valor1 = read_adc(); // Guardar valores de sensor 1
 valor1=valor1*5/255; // Sacando los grados de inclinación
 g1=(valor1-2.5)/1.2;
 g1=asin(g1);
 g1=g1*(180/3.141596); //Conversión de radianes a grados
 m1=g1;
 delay_ms( 100 );
 set_adc_channel(2); // Canal 2 tomo valores para el sensor 2
 delay_ms(20);
 valor2 = read_adc();// Guardar valores de sensor 2
```

```
valor2=valor2*5/255;
g2=(valor2-2.5)/1.2;
g2=asin(g2);
g2=g2*(180/3.141596); //Conversión de radianes a grados
m2=g2;
if (m2>8) // Verifica si el sensor 2 tiene una inclinación mayor a 8°
  {          // Bandera izquierda
d=1;
 i=0;
 }
 else
 {
  if (m2<-8) // Verifica si el sensor 2 tiene una inclinación menor a - 8°
  {             // Bandera derecha
   d=0;
   i=1;
   }
   else
   {
     d=0;
     i=0;
    }
  }
if ( !input (PIN_D2) ) // Verifica si el Pin D2 está en cero
 {
  output_b(0b00000010); // Salida B1 a 1
 }
 else
 {
   if (m1>8 & m1<25) //Verifica si cumple la condición adelante
    {
      if (d==1) // Verifica si bandera está encendida a la derecha
         {
        output_b(0b00110100); // Salida de código derecha-adelante
         }                                B7-B4 = 0011
      else
         {
          if (i==1) // Verifica si bandera está a la izquierda
          {
          output_b(0b01010100); // Salida de código Izquierda-adelante
          }                              // B7-B4 = 0101
          else
          {
           output_b(0b00010100); // Salida de código Adelante
          }                               // B7-B4 = 0001
         }
```

```
      }
      else //e2
      {
       if (m1>-24 & m1<-15) //Verifica si cumple la condición atrás
        {
         if (d==1) // Verifica si bandera está a la derecha
           {
           output_b(0b01000100); // Salida de código derecha-atrás
           }                              // B7-B4 = 0100
         else
           {
            if (i==1) // Verifica si bandera está a la izquierda
            {
            output_b(0b01100100); // Salida de condición izquierda-atrás
            }                              // B7-B4 = 0110
            else
            {
             output_b(0b00100100); // Salida de condición Atrás
            }                              // B7-B4 = 0010
           }
        }
        else //e7
              {
               if (m1>25) //Verifica si condición esta en deshabilitar
                 {
                   output_b(0b01110100); // Salida código deshabilitar
                 }                              // B7-B4 = 0111
               else //e8
                  {
                   if (m2<-15) // Verifica si condición esta en habilitar
                    {
                     output_b(0b10000100); // Salida código habilitar
                    }                              // B7-B4 = 1000
                   else //e9
                    {
                    d=0;
                    i=0;
                    output_b(0b00000100); // En caso que no se cumpla ninguna
                    } //e9                         // condición, Salida de código sin
                  }//e8                          // dirección, B7-B4 = 0000
               }// e7
        } //e2
       } //e1
     delay_ms( 700 );
   }
}
```

### d) Recepción de datos

El siguiente código fuente se encarga de recibir y procesar la información.

```
#include <16F877.h> // Configuración general del microcontrolador
#include <STDLIB.H>
#fuses HS,NOWDT,NOPROTECT
#use delay(clock=10000000)
#byte port_b=6
unsigned int32 contador=0;
unsigned int32 CT=38;

#int_RTCC
RTCC_isr()  // Función para el iniciar el contador
{
 contador++;

 set_rtcc(0);
}
 main()
{
  setup_ccp1(CCP_PWM); // Inicializa el PWM1
  setup_ccp2(CCP_PWM); // Inicializa el PWM2
  setup_timer_2(T2_DIV_BY_4, 255, 1);
  SET_TRIS_B( 0x00 ); // Salidas Puerto B
  SET_TRIS_D( 0XFF); // Entradas Puerto D
  port_b=0;
  output_high(PIN_B0); //Enciende led verde que indica sistema en función normal
loop:  // Etiqueta loop que indica el inicio de la revisión de las posibles decisiones
   //while(1)
  //{
     if ( !input (PIN_D0) ) // Verifica si el voltaje esta fuera del límite permitido
     {
     goto batbaja;  // etiqueta que envía a código de batería baja
     }
     else
     {
    if (!input(PIN_D4) && !input (PIN_D5) && !input (PIN_D6) && input (PIN_D7) )
     {  // Verifica código adelante
      set_pwm1_duty(700); // Activa PWM1
      set_pwm2_duty(0);
```

```
        output_high(PIN_B6);  // Salida B6 = 1
        output_low(PIN_B1);   // Salida B1 = 0
        output_low(PIN_B4);   // Salida B4 = 0
        output_low(PIN_B2);  //  Salida B2 = 0
        goto loop;  // Regresar a etiqueta loop
       }
        else
       {
      if (!input(PIN_D4) && !input (PIN_D5) && input (PIN_D6) && !input (PIN_D7) )
        {  // Verifica código atrás
         set_pwm1_duty(700); // Activa PWM1 69%
         set_pwm2_duty(0);
         output_high(PIN_B1); // Salida B1 = 1
         output_low(PIN_B6); // Salida B6 = 0
         output_low(PIN_B4); // Salida B4 = 0
         output_low(PIN_B2); //  Salida B2 = 0
         goto loop; // Regresar a etiqueta loop
        }
       else
        {
       if (!input(PIN_D4) && !input (PIN_D5) && input (PIN_D6) && input(PIN_D7) )
          { // Verifica código Derecha - Adelante
           set_pwm2_duty(1020); //Activa PWM2 al 100%
           set_pwm1_duty(700);  // Activa PWM1 69%
           output_high(PIN_B4); //Salida B4 = 1
           output_low(PIN_B2); //  Salida B2 = 0
           output_low(PIN_B1); // Salida B1 = 0
           output_high(PIN_B6); // Salida B6 = 1
           goto loop; // Regresar a etiqueta loop
          }
         else
          {
          if (!input(PIN_D4) && input(PIN_D5) && !input(PIN_D6) && !input (PIN_D7) )
//  Verifica código Derecha - Atrás
            {
             set_pwm2_duty(1020); // Activa PWM2 al 100%
             set_pwm1_duty(700);// Activa PWM1 al 69%
             output_low(PIN_B2); //  Salida B2 = 0
             output_high(PIN_B4); //Salida B4 = 1
             output_high(PIN_B1); //Salida B1 = 1
             output_low(PIN_B6); //Salida B6 = 0
             goto loop; // Regresar a etiqueta loop
            }
           else
```

```
          {
           if (!input(PIN_D4) && input(PIN_D5) && !input(PIN_D6) && input (PIN_D7) )
// Verifica código izquierda - adelante
            {
             set_pwm2_duty(1020); // Activa PWM2 al 100%
             set_pwm1_duty(700); // Activa PWM1 al 69%
             output_high(PIN_B2); // Salida B2 = 1
             output_low(PIN_B4); // Salida B4 = 0
             output_low(PIN_B1); // Salida B1 = 0
             output_high(PIN_B6); // Salida B0 = 1
             goto loop; // Regresar a etiqueta loop
            }
           else
            {
           if (!input(PIN_D4) && input(PIN_D5) && input(PIN_D6) && !input (PIN_D7) )
// Verifica código izquierda - atras
             {
              set_pwm2_duty(1020); // Activa PWM2 al 100%
              set_pwm1_duty(700); //Activa PWM1 al 69%
              output_high(PIN_B2); // Salida B2 = 1
              output_low(PIN_B4); // Salida B4 = 0
              output_high(PIN_B1); // Salida B1 = 1
              output_low(PIN_B6); // Salida B6 = 0
              goto loop; // Regresar a etiqueta loop
             }
            else
             {
              if (!input(PIN_D4) && input(PIN_D5) && input(PIN_D6) && input
(PIN_D7) ) //Verifica si código en_pausa
               {
               goto en_pausa; // Regresar a etiqueta en_pausa
               }
                Else // Ejecutar código sin movimiento
                 {
                  set_pwm1_duty(0); //PWM1 0%
                  set_pwm2_duty(0); //PWM1 0%
                  output_low(PIN_B1); // Salida B1 = 0
                  output_low(PIN_B2); // Salida B2 = 0
                  output_low(PIN_B4); // Salida B4 = 0
                  output_low(PIN_B6); // Salida B6 = 0
                  goto loop; // Regresar a etiqueta loop
                 }
               }
             }
```

```
            }
          }
        }
      }
    }
En_pausa:  // Etiqueta que ejecuta las acciones para poner el sistema en_pausa
      // para calcular en periodo de cada interrupción
       ((256-0)*4*256)/10000000=0,0262144 y 1seg / 0,0262144 = 38.14
      CT=38;
      contador=0;
      setup_counters(RTCC_INTERNAL,RTCC_DIV_256);
      enable_interrupts(INT_RTCC);
      enable_interrupts(global);
      set_rtcc(0);
      output_b(0b00000001); //Salida pin B0 = 1 , Led Verde
espera:  // Etiqueta que lleva a código para llevar los segundos del contador
      if(contador>=CT)  // Checa si el contador ya conto 1 segundo
      {
       output_low(PIN_B0); //Apaga led verde
       delay_ms(1000);
       if (contador>=191) // Checa si el contador ya llego al valor deseado de conteo
         {
        CT=38;
        contador=0;  // regresa contador a cero
        set_pwm1_duty(0);
        set_pwm2_duty(0);
        output_low(PIN_B0);
        output_low(PIN_B1);
        output_low(PIN_B2);
        output_low(PIN_B4);
        output_low(PIN_B6);
espera2: // Etiqueta con rutina para verificar código habilitar
          if( input(PIN_D4) && !input(PIN_D5) && !input(PIN_D6) && !input (PIN_D7)
) //Checa código habilitar
            {
            goto habilitar; // Ir a etiqueta con rutina para habilitar sistema
            }
            else
            {
             output_low(PIN_B5); //Pin B5 = 1 enciende led Amarillo, indica sistema
             delay_ms(1000);        // en pausa
             output_high(PIN_B5);
             delay_ms(1000);
            goto espera2;  // Ir a etiqueta para conteo de segundos
```

```
            }
          }
         else
         {
          CT=CT+38;
          output_high(PIN_B0); // Enciende led verde
          delay_ms(1000);
          goto verifica; // Ir a etiqueta verificar
         }
        }
       else
          {
          goto espera; // Ir a etiqueta espera
          }
verifica:  // Etiqueta con código para asegurar si se desea habilitar el sistema
     if(!input(PIN_D4) && input(PIN_D5) && input(PIN_D6) && input (PIN_D7) )
//todavia en habilitar sistema?
      {
       goto espera; // Ir a etiqueta espera
      }
     else
      {
       goto loop; // Ir a etiqueta loop
      }

habilitar: // Etiqueta con rutina para habilitar sistema
       CT=38;
       contador=0;
esperar1:
       if(contador>=CT)  // Verifica si ha pasado un segundo
       {
        if (contador>=191) // Verifica si ya llego al conteo deseado
          {
           CT=38;
           contador=0;
           output_high(PIN_B0); // Enciende led verde
           output_low(PIN_B5); // Apaga led amarillo
           goto loop;  // Ir a etiqueta loop
          }
        else
         {
          CT=CT+38;
          goto verifica2; // Ir a etiqueta verifica2
         }
```

```
       }
      else
         {
          goto esperar1; // Ir a etiqueta esperar1
         }
verifica2:  // Etiqueta que revisa si se desea habilitar el sistema
     if(input(PIN_D4) && !input(PIN_D5) && !input(PIN_D6) && !input (PIN_D7)   )
//todavia en habilitar sistema?
      {
       goto esperar1; // Ir a etiqueta esperar1
      }
     else
      {
       goto espera2; // Ir a etiqueta esperar2
      }
batbaja:  // Etiqueta que ejecuta el código para desactivar sistema por batería baja
       output_low(PIN_B0); // Pin B0 = 0
       output_low(PIN_B1); // Pin B1 = 0
       output_low(PIN_B2); // Pin B2 = 0
       output_low(PIN_B3); // Pin B3 = 0
       output_low(PIN_B4); // Pin B4 = 0
       output_low(PIN_B5); // Pin B5 = 0
       output_low(PIN_B6); // Pin B6 = 0
       output_high(PIN_B7); //Enciende led rojo
espera3:
       goto batbaja;
  }  //fin main
```