```
/*////////////////////////////////////////////////////////////////////////////
 * File:   main.c
 * Author: Emilio Gordillo 18062
 *
 * Seudocodigo
 */

/*////////////////////////////////////////////////////////////////////////////
 * LIBRERIAS
 */
#include <xc.h>
#include <stdint.h>

/*////////////////////////////////////////////////////////////////////////////
 * PIC16F887 Configuration Bit Settings
 */

// CONFIG1
#pragma config FOSC = XT        // Oscillator Selection bits (XT oscillator: Crystal/resonator on RA6/OSC2/CLKOUT and RA7/OSC1/CLKIN)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled and can be enabled by SWDTEN bit of the WDTCON register)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
#pragma config MCLRE = ON       // RE3/MCLR pin function select bit (RE3/MCLR pin function is MCLR)
#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)
#pragma config BOREN = OFF      // Brown Out Reset Selection bits (BOR disabled)
```

```
#pragma config IESO = OFF       // Internal External Switchover bit (Internal/External Switchover mode is disabled)

#pragma config FCMEN = OFF      // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor is disabled)

#pragma config LVP = OFF        // Low Voltage Programming Enable bit (RB3 pin has digital I/O, HV on MCLR must be used for programming)


// CONFIG2

#pragma config BOR4V = BOR40V   // Brown-out Reset Selection bit (Brown-out Reset set to 4.0V)

#pragma config WRT = OFF        // Flash Program Memory Self Write Enable bits (Write protection off)


/*////////////////////////////////////////////////////////////////////////////

 * MACROS

 */

#define _XTAL_FREQ 8000000 //8MHz


/*////////////////////////////////////////////////////////////////////////////

 * VARIABLES

 */

        Char Adc_input[2] = {0b000, 0b001}

        Uint8_t Adc_pin = 0

        char display_array[16] = {0b00111111, 0b00000110, 0b01011011, 0b01001111, 0b01100110, 0b01101101, 0b01111101, 0b00000111, 0b01111111, 0b01101111, 0b01110111, 0b01111100, 0b00111001, 0b01011110, 0b01111001, 0b01110001};

        uint8_t display_count     = 0;

/*////////////////////////////////////////////////////////////////////////////

 * INTERRUPCIONES

 */

void __interrupt () myISR(void){

   }
```

```
   //ADC Interrups
   if (PIR1bits.ADIF == 1 && ADCON0bits.GO_nDONE == 0){
        first_display, second_display = ADC_Read();
        if(Adc_pin == 1){
            adc_pin = 2;
        }
        Else{
            Adc_pin = 1;
        }
        __delay_us(25);
        ADCON0bits.GO_nDONE = 1;
        PIR1bits.ADIF = 0;
/*////////////////////////////////////////////////////////////////////////////
 * PROTOTIPOS DE FUNCIONES
 */
void setup(void) {
        Configurar interrupción TX mediante
        Configurar transmisión Slave
        Activar las GIE
        Modificar el PIE1 y PIR1
        Declarar SYNC como 0
        Declarar CSRC como 0
        Declarar SREN como 1 para recibir
        Declarar CREN como 1 para recibir
        Declarar Spen como 1
        configurar puertos del PIC
/*////////////////////////////////////////////////////////////////////////////
 * CICLO PRINCIPAL
 */
```

```
void main(void) {
   setup();
   setup_ADC();
   while(1){
         PORTD = display_array[display_count];
}
```