Universidad de Valladolid
E.T.S Ingeniería Informática
Grado en Ingeniería Informática

Curso 2018/2019

HARDWARE EMPOTRADO

PRÁCTICA FINAL DE LABORATORIO

Autor:
Sergio Muñumer Blázquez

Introducción

El documento presentado a continuación, expone el diseño hardware y firmware, ideado como solución al problema descrito en el proyecto final de la asignatura “Hardware Empotrado”.

Dicho proyecto se basa en la incorporación de “botones inteligentes” para reducir el cableado en una instalación de ascensores. Estos “botones” serán implementados con microcontroladores PIC10F200.

A mayores se desea implementar un equipo de test de la instalación. Utilizaremos el microcontrolador LPC2103 como “caja de control” del equipo, conectado a una pantalla LCD y un teclado como interfaz para el usuario.

La comunicación entre la caja de control y los botones de los pisos, se realizará mediante un bus serie half-duplex. Forzosamente ha de ser asíncrona, a 38400 bits por segundo, 8 bits de datos y sin paridad.

Los comandos a transmitir son los siguientes:

CMD	Acción	Tecla pulsada
000	Responder con estado	7
001	LED D ON	1
010	LED U ON	2
011	LEDS U+D ON	3
100	Borrar memoria del botón	8
101	LED D OFF	4
110	LED U OFF	5
111	LEDS U+D OFF	6

Diseño Hardware

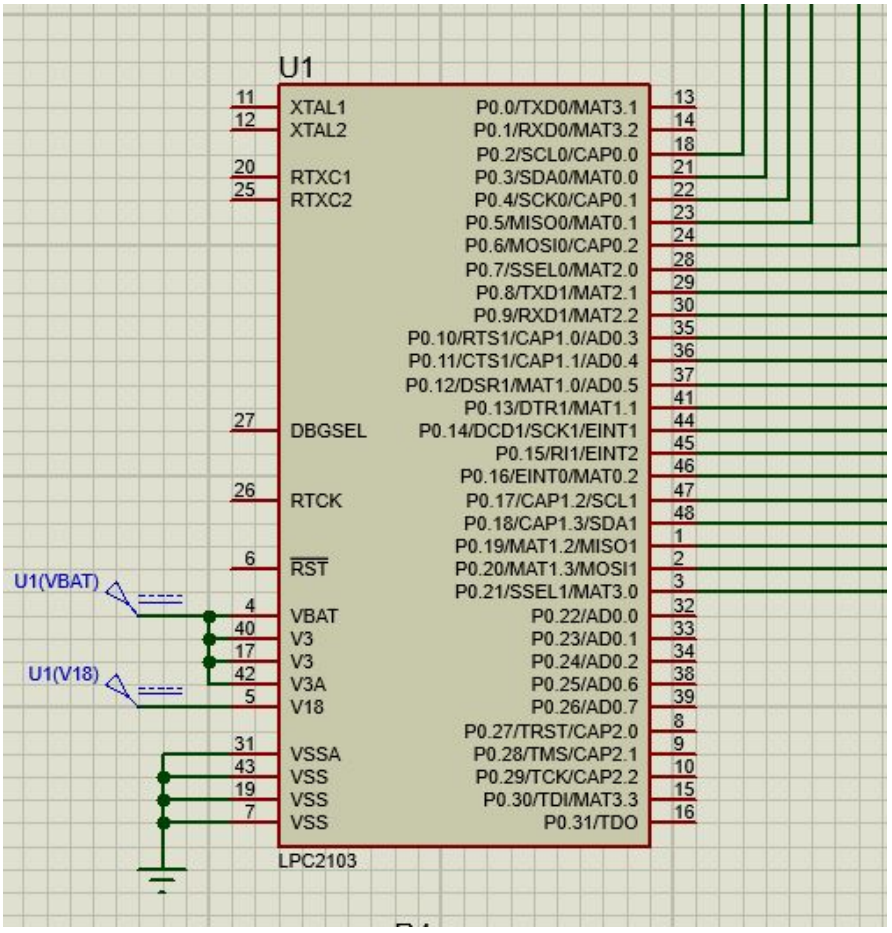
Para comenzar, enumeraremos brevemente los componentes necesarios para el proyecto:

- ❏ Microcontroladores: LPC2103 y PIC10F200.
- ❏ Pantalla LCD - LM016L - 16x2 caracteres.
- ❏ KeyPad -Phone genérico.
- ❏ Resistencias: 10K, 500, 100 y 50 (Ω).
- ❏ LED Rojo, y LED Verde (Uno de cada por cada PIC10F200).

a. Conexiones y cableado

i. LPC2103

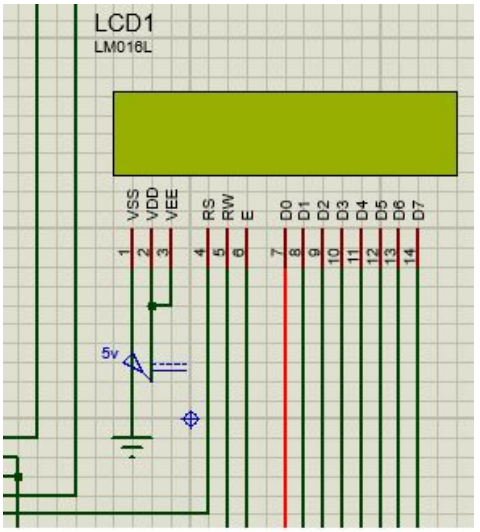
- Alimentación conectada a los pines V18 (1.8V), VBAT, V3 y V3A. (3.3V)
- Conexión a tierra de los pines VSSA, VSS.



- Pines P0.2 - P0.5 como salidas para las filas del KeyPad.
- Pines P0.6, P0.7 y P0.10, como entrada para las columnas del KeyPad.
- Pines P0.8 y P0.9 como salida y entrada del BUS Serie semi-duplex.
- Pines P0.11 a P0.13 señales de control del LCD LM016L
- Pines P0.14 a P0.21 como salidas de datos para la LCD LM016L

ii. LCD-LM016L

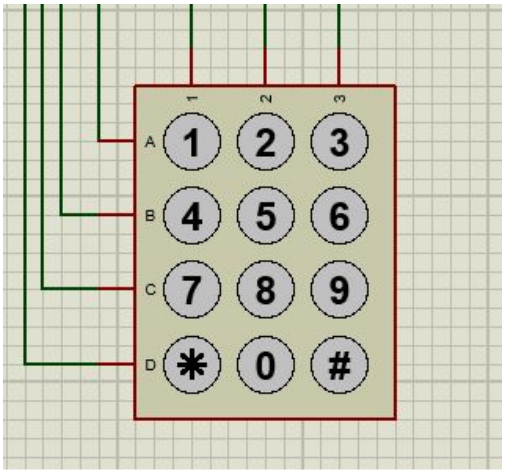
- Pin 1 → Tierra
- Pins 2, 3 → Alimentación 5V
- Pins 4, 5, 6, conectados con P0.11:P0.13 del LPC2103. (RS, RW, E)
- Pins 7-14, conectados con P0.14:P0.21 del LPC2103 (Señal de datos).



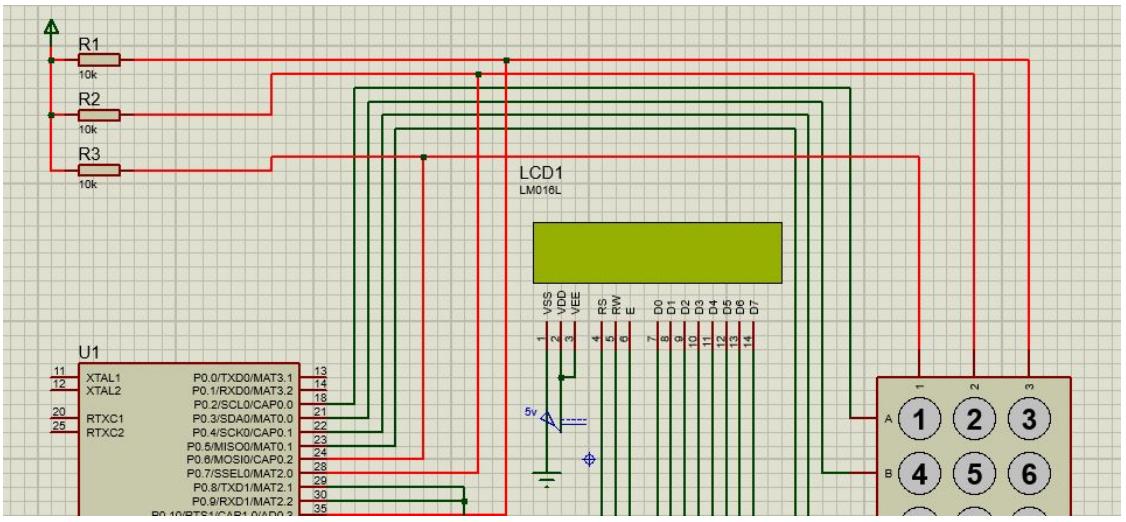
iii. KeyPad-Phone

- Conexión de las filas (A, B, C, D) con los pines del LPC2103 (P0.2 : P0.5)

- Conexión de las columnas (1, 2, 3) con los pines del LPC2103 (P.06, P.07, P.10).

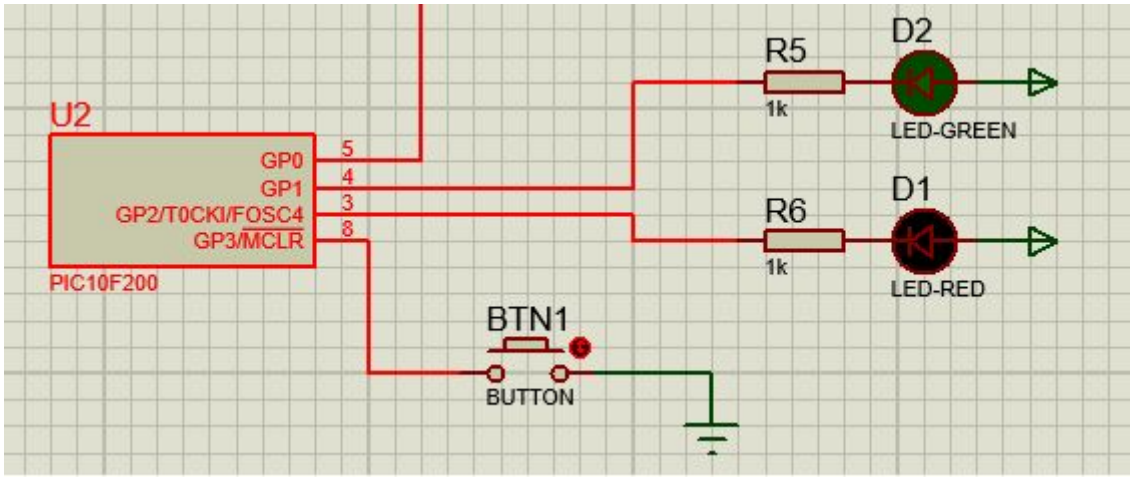


- Implementación de 1 resistencia por cada conexión de las columnas para mantener la tensión alta. (Pull-ups)



iv. PIC10F200

- GP0 → Conexión entrada/salida con el BUS Serie half-duplex.
- GP1 → Conexión saliente hacia un LED verde.
- GP2 → Conexión saliente hacia un LED rojo.
- GP3 → Conexión entrante desde un pulsador conectado a tierra.



Necesitaremos tantos PIC10F200 como pisos del ascensor queramos “simular” en el diseño. Ya que estos son los “botones inteligentes” de las puertas de cada piso.

Diseño Firmware

A continuación, se exponen los scripts necesarios para el proyecto.

Firmware LPC2103

main.c

```
/*****************************************************************************************************
                                        Header files
 *****************************************************************************************************/
#include "lpc21xx.h"
#include "minilib.h"

/*****************************************************************************************************
                                        Definitions
 *****************************************************************************************************/

#define LCD_RS (1<<11)
#define LCD_RW (1<<12)
#define LCD_EN (1<<13)
#define LCD_CLEAR       0x01
#define CURSOR_OFF      0x0C
#define FIRST_ROW       0x80
#define SECOND_ROW     0xC0
#define DELETE_CHAR    0x10
#define SBIT_WordLenght    0x00
#define SBIT_DLAB          0x07
#define SBIT_FIFO          0x00
#define SBIT_RxFIFO        0x01
#define SBIT_TxFIFO        0x02
#define SBIT_RDR           0x00
#define SBIT_THRE          0x05
#define TX1                     16
#define RX1                     18

void lcd_cmd (unsigned char val);
void lcd_data (unsigned char val);
void delay (unsigned int);
void gpio_init(void);
void borrarTecla(void);
void mostrarListado(void);
```

```
char uart_RxChar(void);
void uart_TxChar(char c);
void uart_init(unsigned int c);

/*****************************************************************************************************
                                        Variables
 *****************************************************************************************************/

unsigned char cmd_lcd[5]      = {0x38,0x0e,0x06,0x01,0x80};
unsigned char tester_cmd[8]  = {0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07};
unsigned char pics_dir [2]     = {0x81,0x82};
unsigned char menu0[]          = "Inicio";
unsigned char menu1[]          = "Seleccion";
unsigned char estado0[]        = "B:0 D:0 U:0";
unsigned char estado1[]        = "B:0 D:0 U:1";
unsigned char estado2[]        = "B:0 D:1 U:0";
unsigned char estado3[]        = "B:0 D:1 U:1";
unsigned char estado4[]        = "B:1 D:0 U:0";
unsigned char estado5[]        = "B:1 D:0 U:1";
unsigned char estado6[]        = "B:1 D:1 U:0";
unsigned char estado7[]        = "B:1 D:1 U:1";

char msg1, msg2, respuesta;

int main()
{
unsigned int i,cont=0;
gpio_init();                            // Inicializamos GPIO
uart_init(38400);                       // Inicializamos UART

for (i=0; i<5; i++)                     // Cargamos comandos configuración LCD
        lcd_cmd(cmd_lcd[i]);

while (1) {
        for (i=0; i<7; i++)             // Mensaje: Inicio
                lcd_data(menu0[i]);     // Bucle en espera de una tecla pulsada
        lcd_cmd(SECOND_ROW);    // Cursor LCD en 2º linea
        i=wait_key();                   // Capturamos la tecla pulsada en la variable 'i'

        if(i!=-1){                      // Si 'i' es un valor válido, continuamos
                if((i+49)=='1'){        // Comprobamos si la tecla pulsada es 1
                        lcd_data (i+49);        // Mostramos el valor de la tecla por LCD

                        while (1) {
                        i=wait_key();
```

```
if(i!=-1){
        if((i+49)==':'){            // Continuamos con *
        lcd_data ('*');             // y entramos al menú del PIC
        msg1 = pics_dir[0];     // Cargo  la dirección del PIC
        uart_TxChar(msg1);   // Transmito el primer Byte hacia el PIC

        lcd_cmd(LCD_CLEAR);         // Limpiamos la LCD
        lcd_cmd(SECOND_ROW);    // Cursor en segunda fila LCD
        while (1) {
        i=wait_key();
        if(i!=-1){
        if((i+49)=='1') msg2 = tester_cmd[1]; // Encender D
        if((i+49)=='2') msg2 = tester_cmd[2]; // Encender U
        if((i+49)=='3') msg2 = tester_cmd[3]; // Encender ambos LEDS
        if((i+49)=='4') msg2 = tester_cmd[5]; // Apagar D
        if((i+49)=='5') msg2 = tester_cmd[6]; // Apagar U
        if((i+49)=='6') msg2 = tester_cmd[7]; // Apagar ambos LEDS
        // Consulta el estado_boton
        if((i+49)=='7') msg2 = tester_cmd[0];

        if((i+49)=='8') msg2 = tester_cmd[4]; // Borrar memoria botón
        if((i+49)==';') break;                          // Retorna al menú
        lcd_data(i+49);                     // Mostramos opción en LCD

        // Enviamos el 2º byte con el comando adecuado
        uart_TxChar(msg2);
        // Recibimos la respuesta del esclavo
        respuesta=uart_RxChar();

        lcd_cmd(LCD_CLEAR);         // Limpiamos LCD
/*
En función de la opción seleccionada, el mensaje de respuesta será diferente. En función de la respuesta recibida mostramos por pantalla el resultado.
*/
        if (respuesta == 0x00)
                for (i=0; i<12; i++)
                        lcd_data(estado0[i]);
        if (respuesta == 0x01)
                for (i=0; i<12; i++)
                        lcd_data(estado1[i]);
        if (respuesta == 0x02)
                for (i=0; i<12; i++)
                        lcd_data(estado2[i]);
        if (respuesta == 0x03)
                for (i=0; i<12; i++)
```

```
                        lcd_data(estado3[i]);
                if (respuesta == 0x04)
                        for (i=0; i<12; i++)
                                lcd_data(estado4[i]);
                if (respuesta == 0x05)
                        for (i=0; i<12; i++)
                                lcd_data(estado5[i]);

                if (respuesta == 0x06)
                        for (i=0; i<12; i++)
                                lcd_data(estado6[i]);
                if (respuesta == 0x07)
                        for (i=0; i<12; i++)
                                lcd_data(estado7[i]);
        }
        break;          // Salimos del bucle while interno
        }
}
}
break;                  // Salimos del bucle while externo
}       //fin while
}       //fin if
if((i+49)=='2'){        // Comprobamos si la tecla pulsada es 1
        lcd_data (i+49);        // Mostramos el valor de la tecla por LCD

        while (1) {
        i=wait_key();
        if(i!=-1){
                if((i+49)==':'){            // Continuamos con *
                lcd_data ('*');             // y entramos al menú del PIC
                msg1 = pics_dir[1];     // Cargo  la dirección del PIC
                uart_TxChar(msg1);  // Transmito el primer Byte hacia el PIC

                lcd_cmd(LCD_CLEAR);         // Limpiamos la LCD
                lcd_cmd(SECOND_ROW);    // Cursor en segunda fila LCD
                while (1) {
                i=wait_key();
                if(i!=-1){
                if((i+49)=='1') msg2 = tester_cmd[1]; // Encender D
                if((i+49)=='2') msg2 = tester_cmd[2]; // Encender U
                if((i+49)=='3') msg2 = tester_cmd[3]; // Encender ambos LEDS
                if((i+49)=='4') msg2 = tester_cmd[5]; // Apagar D
                if((i+49)=='5') msg2 = tester_cmd[6]; // Apagar U
                if((i+49)=='6') msg2 = tester_cmd[7]; // Apagar ambos LEDS
                // Consulta el estado botón
                if((i+49)=='7') msg2 = tester_cmd[0];
```

```
        if((i+49)=='8') msg2 = tester_cmd[4]; // Borrar memoria botón
        if((i+49)==';') break;                          // Retorna al menú
        lcd_data(i+49);                 // Mostramos opción en LCD

        // Enviamos el 2º byte con el comando adecuado
        uart_TxChar(msg2);
        // Recibimos la respuesta del esclavo
        respuesta=uart_RxChar();

        lcd_cmd(LCD_CLEAR);         // Limpiamos LCD
/*
En función de la opción seleccionada, el mensaje de respuesta será diferente. En función de la respuesta recibida mostramos por pantalla el resultado.
*/
if(((i+49)=='1') || ((i+49)=='6') || ((i+49)=='7')){

        if (respuesta == 0x00)
                for (i=0; i<12; i++)
                        lcd_data(estado0[i]);
        if (respuesta == 0x01)
                for (i=0; i<12; i++)
                        lcd_data(estado1[i]);
        if (respuesta == 0x02)
                for (i=0; i<12; i++)
                        lcd_data(estado2[i]);
        if (respuesta == 0x03)
                for (i=0; i<12; i++)
                        lcd_data(estado3[i]);
        if (respuesta == 0x04)
                        for (i=0; i<12; i++)
                                lcd_data(estado4[i]);
        if (respuesta == 0x05)
                for (i=0; i<12; i++)
                        lcd_data(estado5[i]);

        if (respuesta == 0x06)
                for (i=0; i<12; i++)
                        lcd_data(estado6[i]);
        if (respuesta == 0x07)
                for (i=0; i<12; i++)
                        lcd_data(estado7[i]);
        }
}
break;          // Salimos del bucle while interno
}
```

```
                }
                }
                break;                  // Salimos del bucle while externo
                }       //fin while
                }       //fin if
                if((i+49)=='<') borrarTecla();
                else if((i+49)==':') mostrarListado();


                else{ lcd_cmd(LCD_CLEAR);} // Limpiamos pantalla
                }
        }
}

/*
MÉTODO PARA BORRAR LA ÚLTIMA TECLA PULSADA
RETROCESO DE CURSOR + INSERTAR ESPACIO
*/
void borrarTecla()
{
        char comando = 0x10;
        lcd_cmd(comando);
        lcd_data(32);
        lcd_cmd(comando);
}

/*
MÉTODO PARA MOSTRAR EL LISTADO DE DIRECCIONES
DISPONIBLES (No implementado)
*/
void mostrarListado()
{
        lcd_cmd(LCD_CLEAR);
}

/*
MÉTODO PARA INICIAR LA INTERFAZ GPIO
*/
void gpio_init()
{
        PINSEL0 = 0x00000000;
        IO0DIR = 0x003FF83C;
        IO0CLR = 0x003FF83C;
}

/*
```

```
MÉTODO PARA ENVIAR UN COMANDO A LA LCD
*/
void lcd_cmd(unsigned char val)
{
        IO0PIN = (val<<14);
        IO0CLR = LCD_RS;
        IO0CLR = LCD_RW;
        IO0SET = LCD_EN;
        _delay_ms(50);
        IO0CLR = LCD_EN;
        _delay_ms(100);

}
/*
MÉTODO PARA ENVIAR UN CARÁCTER A LA LCD
*/
void lcd_data(unsigned char val)
{

        IO0PIN = (val<<14);
        IO0SET = LCD_RS;
        IO0CLR = LCD_RW;
        IO0SET = LCD_EN;
        _delay_ms(50);
        IO0CLR = LCD_EN;
        _delay_ms(100);

}

/*
Teclado. Conexionado:
Filas: P0.2 a P0.5 Columnas: P0.6 + P0.7 + P0.10
Se retorna -1 si no hay ninguna tecla pulsada
*/
int scan_key()
{
int fila,col,d;
IO0CLR=0x3C;                    // latches de filas en cero

for (fila=0;fila<4;fila++) {
// simulamos colector abierto actuando sobre la
// dirección de los pines de fila
IO0DIR=(IO0DIR&0xFFFFFFC3)|(1<<(fila+2));

_delay_ms(5); // espera
d=((~IO0PIN)>>6)&0x3; // estado de las columnas
```

```
        if (d) { // búsqueda de columna activa
                for (col=0;col<3;col++) {
                        if (d&1) return fila*3+col;d>>=1;

                        }
                }
d=((~IO0PIN)>>10)&0x1; // estado de las columnas
        if (d) { // búsqueda de columna activa
                if (d&1) return fila*3+2;
        }

}

return -1;
}

/*
Teclado. Esperamos por pulsación
*/
int wait_key()
{
int key;
// esperamos que se suelten las teclas pulsadas
while (scan_key()>=0) _delay_ms(20);
// esperamos que se pulse una tecla
while ((key=scan_key())<0) _delay_ms(20);
return key;
}

/*
MÉTODO PARA INICIAR LA INTERFAZ UART
RX1 & TX1
*/
void uart_init(unsigned int baudrate)
{
   unsigned int var_RegValue_u32;

   PINSEL0 |= (1<<RX1) | (1<<TX1);
        // Configuramos P0.8/P0.9 como RX1 y TX1
   U1FCR = (1<<SBIT_FIFO) | (1<<SBIT_RxFIFO) | (1<<SBIT_TxFIFO);
        // Activados: FIFO & reset Rx/Tx FIFO buffers
   U1LCR = (0x03<<SBIT_WordLenght) | (1<<SBIT_DLAB);
        // 8bit datos, 1Stop bit, No paridad
```

```
    var_RegValue_u32 = ( PCLK / (16 * baudrate ));
    U1DLL =  var_RegValue_u32 & 0xFF;
    U1DLM = (var_RegValue_u32 >> 0x08) & 0xFF;

          U1LCR= 0x03;
}

/*
MÉTODO PARA TRANSMITIR 1 CARACTER
*/
void uart_TxChar(char ch)
{
          U1THR = ch;
          while( (U1LSR & 0x40) == 0 );
}


/*
METODO PARA RECIBIR 1 CARÁCTER
*/
char uart_RxChar()
{
 char ch;
          while( (U1LSR & 0x01) == 0);
    ch = U1RBR;
 return ch;
}
```

PIC10F200 (U2)

El código de cada PIC10F200 está replicado, solo variando la 3º línea del loop principal del programa. En dicha línea se compara con una máscara, que contiene el valor de su dirección. Así el PIC número 1, posee la dirección 0x81, el PIC 2 la 0x82 y así sucesivamente.

main.asm

```
;=============================================================================
; Main.asm file generated by New Project wizard
;
; Created:   lu. may. 20 2019
; Processor: PIC10F200
; Compiler:  MPASM (Proteus)
;=============================================================================


#include p10f200.inc          ; Include register definition file
  __config 0x000              ;
  radix dec                   ; números en decimal por defecto

;=============================================================================
;                                DEFINITIONS
;=============================================================================


;=============================================================================
;                                VARIABLES
;=============================================================================
  cblock 0x10                       ;Primera posición libre
  CNT                               ;CONTADOR DE RETARDOS
  CNTBIT                            ;CONTADOR DE BITS
  DATO                              ;DATO A TRANSMITIR
  COMANDO                           ;COMANDO RECIBIDO
  RESPUESTA                         ;RESPUESTA A TRANSMITIR
  COUNT                             ;CONTADOR
  BOTONMEMORY                       ;MEMORIA DEL BOTÓN

  endc
```

```
;*****************************************************************
;                     RESET and INTERRUPT VECTORS
;*****************************************************************

; Reset Vector
RST   code  0x0

   ANDLW 0xFE
   MOVWF OSCCAL                    ;Calibramos oscilador
   MOVLW 0x08                      ;Iniciamos gpio a '1000' GPIO3 input
   TRIS GPIO                       ;GPIO0/GPIO1/GPIO2 outputs
   BSF GPIO, 2                     ;Apago led D
   BSF GPIO, 1                     ;Apago led U
   MOVLW 0x0F                      ;T0CS OFF, Prescaler 1/128 para WDT
   OPTION
   CLRW
   GOTO loop_main



;*****************************************************************
;                        CODE SEGMENT
;*****************************************************************

loop_main:


   CALL RXBYTE                     ;guarda byte en PREGUNTA
   MOVF COMANDO, W                 ;Muevo PREGUNTA a W
   XORLW 0x81                      ;Calculo la XOR entre el valor de W y 0x81
   BTFSS STATUS, 2                 ;Compruebo si son iguales fijandome en el flag Z
   GOTO loop_main                  ;si no son iguales, vuelva a empezar

   ;Calculo estado del PIC

   MOVF BOTONMEMORY, W
   MOVWF RESPUESTA
   RLF RESPUESTA, F                ;Roto para almacenar el siguiente valor

   MOVLW 0                         ;Cargo 0 en W
   BTFSC GPIO, 1                   ;Compruebo el gpio1, si esta a 0 salto la siguiente
   MOVLW 1                         ;si esta a 1 cargo 1 en W
   IORWF RESPUESTA, F              ;hago una OR entre W y RESPUESTA y guardo
   RLF RESPUESTA, F                ;Roto para almacenar el siguiente valor

   MOVLW 0                         ;Cargo 0 en W
```

```
BTFSC GPIO, 2              ;Compruebo el gpio2 si esta a 0 salto la siguiente
MOVLW 1                    ;si esta a 1 cargo 1 en W
IORWF RESPUESTA, F         ;hago una OR entre W y RESPUESTA y guardo

CALL RXBYTE                ;captura del byte
MOVF RESPUESTA, W          ;devuelvo estado a W
CALL  TXBYTE               ;TXBYTE transmite el valor en W

;A continuacion se compara el comando recibido con diferentes mascaras
;para asi localizar la orden recibida y ejecutar el metodo adecuado

MOVF COMANDO, W            ;transmitimos el byte guardado en COMANDO a W
BCF STATUS, 2              ;limpiamos flag Z
XORLW 0x00                 ;comparo para ver si es el comando 0x00 el recibido
BTFSC STATUS, 2            ;Compruebo si son iguales fijandome en el flag Z
CALL        TX7

MOVF COMANDO, W
BCF STATUS, 2
XORLW 0x01
BTFSC STATUS, 2
CALL        TX1

MOVF COMANDO, W
BCF STATUS, 2
XORLW 0x02
BTFSC STATUS, 2
CALL        TX2

MOVF COMANDO, W
BCF STATUS, 2
XORLW 0x03
BTFSC STATUS, 2
CALL        TX3

MOVF COMANDO, W
BCF STATUS, 2
XORLW 0x05
BTFSC STATUS, 2
CALL        TX4

MOVF COMANDO, W
BCF STATUS, 2
XORLW 0x06
BTFSC STATUS, 2
CALL        TX5
```

```
  MOVF COMANDO, W
  BCF STATUS, 2
  XORLW 0x07
  BTFSC STATUS, 2
  CALL          TX6

  MOVF COMANDO, W
  BCF STATUS, 2
  XORLW 0x04
  BTFSC STATUS, 2
  CALL          TX8


  BSF GPIO, 0
  GOTO loop_main

FIN:
    goto FIN

 ;=====================================================================
 ;                          METODO PARA ENCENDER  LED D
 ;=====================================================================
TX1:

  BCF GPIO, 2
  BSF GPIO, 1

  MOVF BOTONMEMORY, W
  MOVWF RESPUESTA
  RLF RESPUESTA, F        ;Roto  para almacenar el siguiente valor

  MOVLW 0                 ;Cargo 0 en W
  BTFSC GPIO, 1           ;Compruebo el gpio1, si esta a 0 salto la siguiente
  MOVLW 1                 ;si esta a 1 cargo 1 en W
  IORWF RESPUESTA, F      ;hago una OR entre W y RESPUESTA y guardo
  RLF RESPUESTA, F        ;Roto para almacenar el siguiente valor

  MOVLW 0                 ;Cargo 0 en W
  BTFSC GPIO, 2           ;Compruebo el gpio2 si esta a 0 salto la siguiente
  MOVLW 1                 ;si esta a 1 cargo 1 en W
  IORWF RESPUESTA, F      ;hago una OR entre W y RESPUESTA y guardo
  RETLW 0
```

```
;=================================================================
;                        METODO PARA ENCENDER  LED U
;=================================================================

TX2:

  BCF GPIO, 1

  MOVF BOTONMEMORY, W
  MOVWF RESPUESTA
  RLF RESPUESTA, F        ;Roto para almacenar el siguiente valor

  MOVLW 0                 ;Cargo 0 en W
  BTFSC GPIO, 1           ;Compruebo el gpio1, si esta a 0 salto la siguiente
  MOVLW 1                 ;si esta a 1 cargo 1 en W
  IORWF RESPUESTA, F      ;hago una OR entre W y RESPUESTA y guardo
  RLF  RESPUESTA, F       ;Roto para almacenar el siguiente valor

  MOVLW 0                 ;Cargo 0 en W
  BTFSC GPIO, 2           ;Compruebo el gpio2 si esta a 0 salto la siguiente
  MOVLW 1                 ;si esta a 1 cargo 1 en W
  IORWF RESPUESTA, F      ;hago una OR entre W y RESPUESTA y guardo

  RETLW 0

;=================================================================
;                     MÉTODO PARA ENCENDER AMBOS LEDS U+S
;=================================================================

TX3:

  BCF GPIO, 2
  BCF GPIO, 1

  MOVF BOTONMEMORY, W
  MOVWF RESPUESTA
  RLF RESPUESTA, F        ;Roto para almacenar el siguiente valor

  MOVLW 0                 ;Cargo 0 en W
  BTFSC GPIO, 1           ;Compruebo el gpio1, si esta a 0 salto la siguiente
  MOVLW 1                 ;si esta a 1 cargo 1 en W
  IORWF RESPUESTA, F      ;hago una OR entre W y RESPUESTA y guardo
```

```
  RLF RESPUESTA, F      ;Roto para almacenar el siguiente valor

  MOVLW 0               ;Cargo 0 en W
  BTFSC GPIO, 2         ;Compruebo el gpio2 si esta a 0 salto la siguiente
  MOVLW 1               ;si esta a 1 cargo 1 en W
  IORWF RESPUESTA, F    ;hago una OR entre W y RESPUESTA y guardo

  RETLW 0

 ;=====================================================================
 ;                          METODO PARA APAGAR LED D
 ;=====================================================================
TX4:

  BSF GPIO, 2

  MOVF BOTONMEMORY, W
  MOVWF RESPUESTA
  RLF RESPUESTA, F      ;Roto para almacenar el siguiente valor

  MOVLW 0               ;Cargo 0 en W
  BTFSC GPIO, 1         ;Compruebo el gpio1, si esta a 0 salto la siguiente
  MOVLW 1               ;si esta a 1 cargo 1 en W
  IORWF RESPUESTA, F    ;hago una OR entre W y RESPUESTA y guardo
  RLF RESPUESTA, F      ;Roto para almacenar el siguiente valor

  MOVLW 0               ;Cargo 0 en W
  BTFSC GPIO, 2         ;Compruebo el gpio2 si esta a 0 salto la siguiente
  MOVLW 1               ;si esta a 1 cargo 1 en W
  IORWF RESPUESTA, F    ;hago una OR entre W y RESPUESTA y guardo

  RETLW 0

 ;=====================================================================
 ;                          METODO PARA APAGAR LED U
 ;=====================================================================
TX5:

  BSF GPIO, 1

  MOVF BOTONMEMORY, W
  MOVWF RESPUESTA
  RLF RESPUESTA, F      ;Roto para almacenar el siguiente valor

  MOVLW 0               ;Cargo 0 en W
  BTFSC GPIO, 1         ;Compruebo el gpio1, si esta a 0 salto la siguiente
```

```
  MOVLW 1                 ;si esta a 1 cargo 1 en W
  IORWF RESPUESTA, F      ;hago una OR entre W y RESPUESTA y guardo
  RLF RESPUESTA, F        ;Roto para almacenar el siguiente valor

  MOVLW 0                 ;Cargo 0 en W
  BTFSC GPIO, 2           ;Compruebo el gpio2 si esta a 0 salto la siguiente
  MOVLW 1                 ;si esta a 1 cargo 1 en W
  IORWF RESPUESTA, F      ;hago una OR entre W y RESPUESTA y guardo

  RETLW 0


;=================================================================
;                   METODO PARA APAGAR LOS 2 LEDS U+D
;=================================================================


TX6:

  MOVLW 0xE                     ;Diferentes formas probadas de manejar GPIO
  TRIS GPIO

  MOVF BOTONMEMORY, W
  MOVWF RESPUESTA
  RLF RESPUESTA, F        ;Roto para almacenar el siguiente valor

  MOVLW 0                 ;Cargo 0 en W
  BTFSC GPIO, 1           ;Compruebo el gpio1, si esta a 0 salto la siguiente
  MOVLW 1                 ;si esta a 1 cargo 1 en W
  IORWF RESPUESTA, F      ;hago una OR entre W y RESPUESTA y guardo
  RLF RESPUESTA, F        ;Roto para almacenar el siguiente valor

  MOVLW 0                 ;Cargo 0 en W
  BTFSC GPIO, 2           ;Compruebo el gpio2 si esta a 0 salto la siguiente
  MOVLW 1                 ;si esta a 1 cargo 1 en W
  IORWF RESPUESTA, F      ;hago una OR entre W y RESPUESTA y guardo


  RETLW 0
```

```
;=====================================================================
;                    MÉTODO PARA CAPTURAR EL ESTADO DEL PIC
;=====================================================================
TX7:

  MOVF BOTONMEMORY, W
  MOVWF RESPUESTA
  RLF RESPUESTA, F        ;Roto para almacenar el siguiente valor

  MOVLW 0                 ;Cargo 0 en W
  BTFSC GPIO, 1           ;Compruebo el gpio1, si esta a 0 salto la siguiente
  MOVLW 1                 ;si esta a 1 cargo 1 en W
  IORWF RESPUESTA, F      ;hago una OR entre W y RESPUESTA y guardo
  RLF RESPUESTA, F        ;Roto  para almacenar el siguiente valor

  MOVLW 0                 ;Cargo 0 en W
  BTFSC GPIO, 2           ;Compruebo el gpio2 si esta a 0 salto la siguiente
  MOVLW 1                 ;si esta a 1 cargo 1 en W
  IORWF RESPUESTA, F      ;hago una OR entre W y RESPUESTA y guardo

  RETLW 0

;=====================================================================
;                    METODO PARA BORRAR LA PULSACION DEL BOTON
;=====================================================================
TX8:
  ;Limpiamos la posicion de memoria BOTONMEMORY
  CLRF BOTONMEMORY

  ;Calculamos el estado
  MOVF BOTONMEMORY, W
  MOVWF RESPUESTA
  RLF RESPUESTA, F        ;Roto para almacenar el siguiente valor

  MOVLW 0                 ;Cargo 0 en W
  BTFSC GPIO, 1           ;Compruebo el gpio1, si esta a 0 salto la siguiente
  MOVLW 1                 ;si esta a 1 cargo 1 en W
  IORWF RESPUESTA, F      ;hago una OR entre W y RESPUESTA y guardo
  RLF RESPUESTA, F        ;Roto para almacenar el siguiente valor

  MOVLW 0                 ;Cargo 0 en W
  BTFSC GPIO, 2           ;Compruebo el gpio2 si esta a 0 salto la siguiente
  MOVLW 1                 ;si esta a 1 cargo 1 en W
  IORWF RESPUESTA, F      ;hago una OR entre W y RESPUESTA y guardo
```

```
  RETLW 0

;=====================================================================
;                    MÉTODO PARA TRANSMITIR UN BYTE POR GPIO0
;=====================================================================
TXBYTE:

    MOVWF DATO              ;Guardamos W en DATO
    MOVLW  8                        ;Preparamos el contador de bits
    MOVWF CNTBIT
    BCF GPIO,0              ;Comprobamos si GPIO0 ESTA A 0
    MOVLW 7                 ;Retardo para el stop nicial
    MOVWF CNT

loop_reetardo:DECFSZ CNT , F
                  GOTO loop_reetardo

txbit:
    MOVF DATO,W             ;Movemos DATO a W
    MOVWF GPIO              ;Y W a GPIO
    RRF DATO, F             ;Rotamos dato a la derecha para el siguiente bit
    MOVLW 6                 ;Pausa
    MOVWF CNT

loop_retardo2: DECFSZ CNT, F
                  GOTO loop_retardo2

    nop
    DECFSZ CNTBIT ,F        ;Decrementamos contador de bits
    GOTO txbit              ;Volvemos a txbit por el siguiente bit
    BSF GPIO,0              ;Comprobamos si GPIO0 esta a 1
    MOVLW 6                 ;Stop final
    MOVWF CNT

loop_retardo3: DECFSZ CNT, F
                  GOTO loop_retardo3


    RETLW 0                 ;Retorno de la rutina
```

```
;=================================================================
;                   METODO PARA RECIBIR UN BYTE POR GPIO0
;=================================================================
RXBYTE:

    MOVLW 0x01

    BTFSS GPIO, 3           ;Compruebo si son iguales fijandome en el flag Z
    IORWF BOTONMEMORY, f    ;Operación OR - memoria del botón
    BTFSC GPIO,0            ;Comprobamos si GPIO0 esta a 1

    GOTO RXBYTE             ;Sino lo esta, volvemos a empezar, si está seguimos
    MOVLW  8                        ;Preparamos el número de bits para recibir
    MOVWF CNTBIT            ;Guardamos en CNTBIT

    CLRF COMANDO            ;Limpio COMANDO en memoria para borrar el anterior
    CALL half_baud          ;Invoco retardos para comprobar el stop inicial
    BTFSC GPIO,0            ;Compruebo, si está a 1 seguimos sino, era un error
    GOTO RXBYTE             ;Retorno al inicio

RXBIT:

    CALL baud                       ;Invoco retardos
    BCF STATUS,0                    ;Comprobamos el flag de la ALU
    RRF COMANDO, F                  ;Rotamos a la derecha COMANDO
    BTFSC GPIO, 0                   ;Comprobamos si GPIO es 1
    BSF COMANDO, 7                  ;Pongo 1 en la ultima posicion de COMANDO
    DECFSZ CNTBIT ,F                ;Decrementamos el contador de bits

    GOTO RXBIT                      ;Volvemos a RXBIT para recibir siguiente bit
    CALL baud                       ;Esperamos el Stop final

    MOVF COMANDO, W         ;Movemos el COMANDO recibido  a W


    RETLW 0                         ;Retorno de la rutina
```

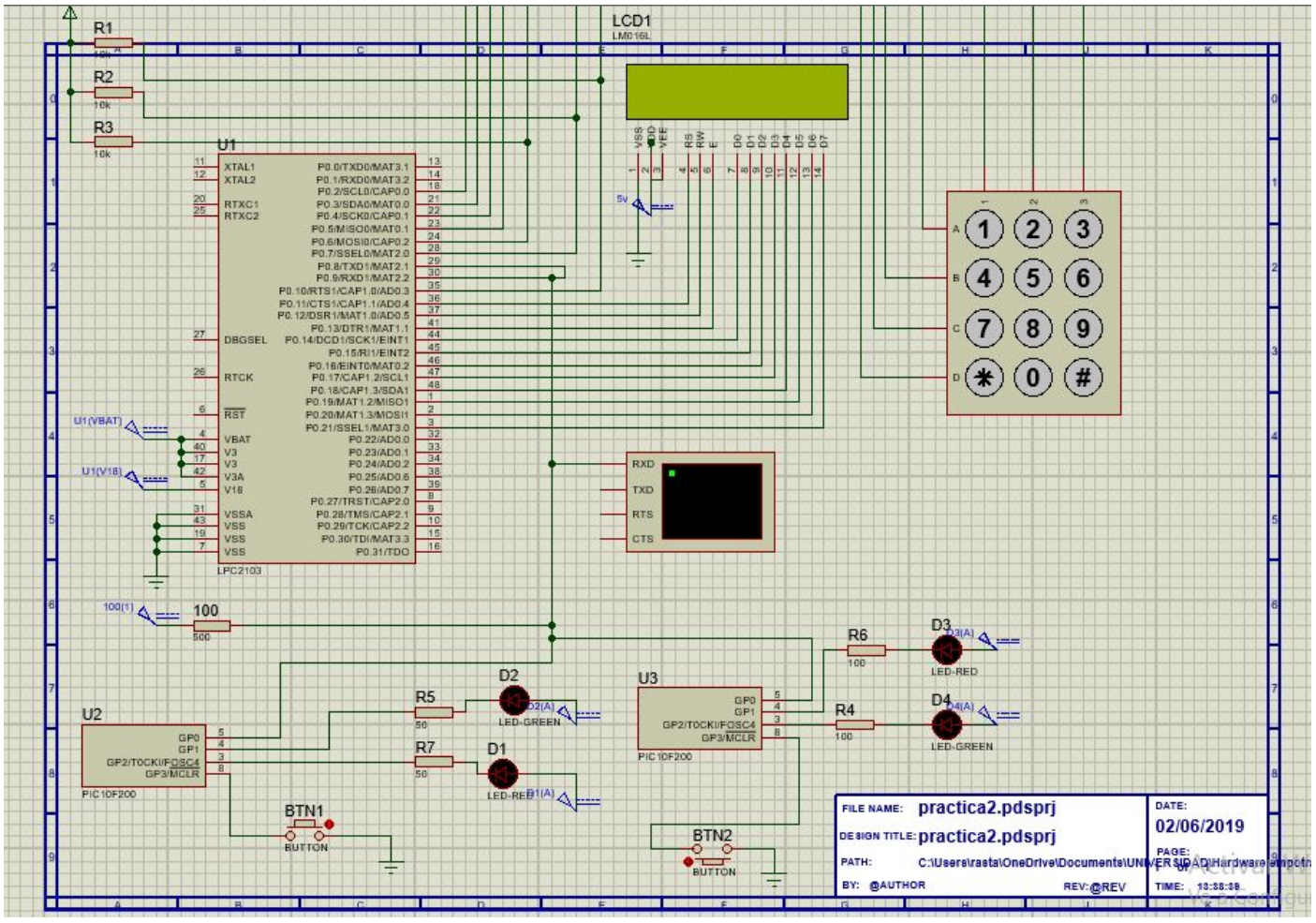
```
 ;=====================================================================
 ;                          RETARDOS PARA 38400 BAUDIOS
 ;=====================================================================
baud:
    MOVLW D'2'
    MOVWF COUNT
baud1:
    DECFSZ COUNT,F
    GOTO baud1
half_baud:
    MOVLW D'2'
    MOVWF COUNT
hbaud1:
    DECFSZ COUNT,F
    GOTO hbaud1

    RETLW 0

;=====================================================================
```

Anexo

Captura de pantalla con el diseño.



Bibliografía

[1] MicroChip Forums Website

[2] PIC10F200 Datasheet

[3] Material del profesor

[4] The Embedded Lab

[5] LPC2103 Datasheet