

# Microcontroladores

## Laboratorio Sesión 9

Semestre: 2022-1

Profesor: Kalun José Lau Gan

1

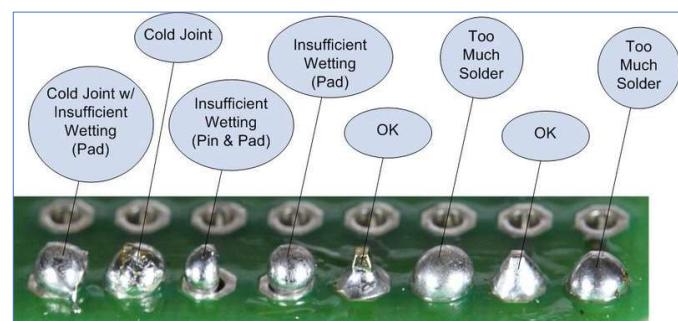
### Preguntas previas:

- ¿Cómo soldar bien?

- Tener buenas herramientas: cautín con control de temperatura, pinzas, pelacables, desarmadores, **pasta para soldar (flux)**, soldadura de buena calidad, esponja humedecida con agua, alcohol isopropílico.
- Practicar y desarrollar la habilidad (mirar videos instruccionales)
- Verificar que lo soldado tenga una forma cónica, si esta en circunferencia significa que es un soldado frío.

- ¿Empresas que desarrollan PCBs?

- Jlcpcb.com
- Pcbway.com



2

## Agenda:

- Aspectos introductorios del XC8
- El display LCD alfanumérico HD44780
- Librería LCD para XC8
- El módulo conversor A/D en XC8

3

## Referencia de plantilla:

- La plantilla para los códigos en C se ha basado en la plantilla de programas en Arduino IDE:
  - Uso de función `setup()` donde se coloca los aspectos iniciales de configuración antes de correr el programa de la aplicación. En XC8 crearemos una función similar. Por ejemplo `configuración()`
  - Uso de función `loop()` donde se detalla el programa de la aplicación. En XC8 usaremos la función `main()` donde dentro llamaremos a la función `configuración` antes de detallar el programa de usuario.

```

four_steps_arduino_program_template | Arduino 1.8.5

four_steps_arduino_program_template

/*
 *Title: My Awesome Arduino Program
 *Author: Liz Miller
 *Date: 02/15/2018
 *Version: v1.0
 *Purpose: This code shows you how to write an Arduino Program!
 */
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}

Done Saving.

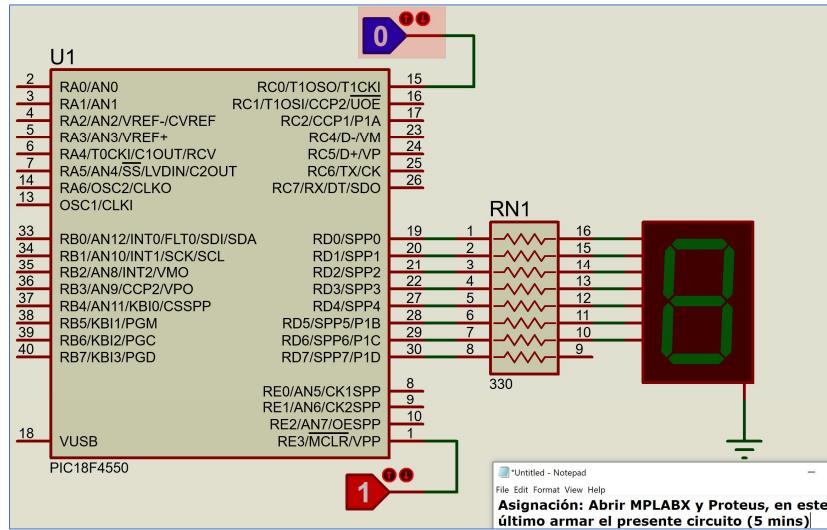
8 Arduino/Genuino Uno on /dev/cu.usbmodem1421

```

4

## Ejemplo: Visualizar “HOLA UPC”

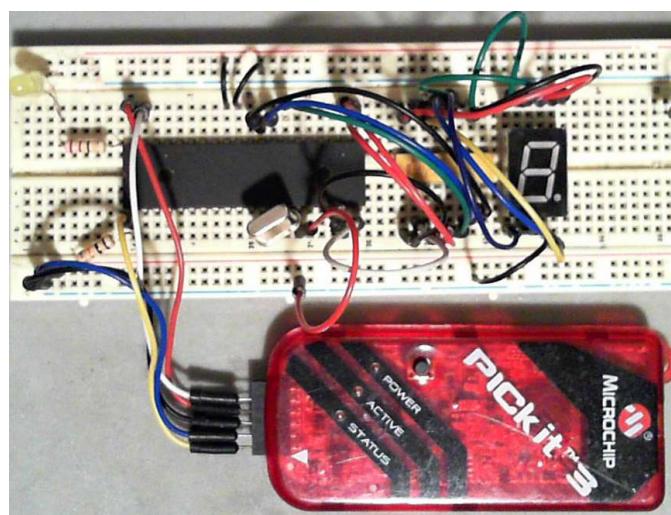
- Si  $RC0=1$  la visualización del mensaje saldrá de cabeza



5

## Ejemplo: Visualizar “HOLA UPC”

- Si  $RC0=1$  la visualización del mensaje saldrá de cabeza



6

## Ejemplo: Visualizar “HOLA UPC”

- Prototipo visualizando el mensaje empleando LUT (lookup table)

```

1  #include <xc.h>
2  #include "cabecera.h"
3  #define _XTAL_FREQ 48000000UL
4
5  //Declaracion de variables globales
6  unsigned char cuenta = 0;
7  //unsigned char tabla_7s[] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x67};
8  unsigned char holaupc[] = {0x76, 0x3F, 0x38, 0x77, 0x00, 0x3E, 0x73, 0x39, 0x00};
9  //          H   O   L   A   U   P   C
10
11 void configuro(void) {
12     TRISD = 0x80; //RD0:RD6 son salidas
13 }
14
15 void main(void) {
16     configuro();
17     while(1){
18         LATD = holaupc[cuenta];
19         __delay_ms(200);
20         if (cuenta == 8){
21             cuenta = 0;
22         }
23         else{
24             cuenta++;
25         }
26     }
27 }
```

7

## Ejemplo: Visualizar “HOLA UPC”

- Análisis para obtener el algoritmo de colocar las letras de cabeza ( voltear 180°)

Tenemos la letra A:

$\{ \begin{array}{l} \text{temp1} = D1 \ll 3 \\ \text{temp1} = \text{temp1} \& 0x38 \end{array}$

$\{ \begin{array}{l} \text{temp2} = D1 \gg 3 \\ \text{temp2} = \text{temp2} \& 0x07 \end{array}$

$\{ \text{temp3} = D1 \& 0x40$

$D2 = \text{temp1} + \text{temp2} + \text{temp3}$

$c = ?$

```

11 unsigned char de_cabeza(unsigned char dato){
12     unsigned char temp1 = 0;
13     unsigned char temp2 = 0;
14     unsigned char temp3 = 0;
15     unsigned char salida = 0;
16     temp1 = dato << 3;
17     temp1 = temp1 & 0x38;
18     temp2 = dato >> 3;
19     temp2 = temp2 & 0x07;
20     temp3 = dato & 0x40;
21     salida = temp1 + temp2 + temp3;
22     return salida;
23 }
```

Temp1	0 0 c b a 0 0 0	+
Temp2	0 0 0 0 0 f e d	
Temp3	0 1 0 0 0 0 0 0	
	<hr/>	
	0 0 c b a f e d	

8

## Ejemplo: Visualizar “HOLA UPC”

- Código final

```

1  #include <xc.h>
2  #include "cabecera.h"
3  #define _XTAL_FREQ 48000000UL
4
5  //Declaracion de variables globales
6  unsigned char cuenta = 0;
7  //unsigned char tabla_7s[] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7E, 0x67};
8  unsigned char holaupc[] = {0x76, 0x3F, 0x38, 0x77, 0x00, 0x3E, 0x73, 0x39, 0x00};
9  //
10
11 void configuro(void){
12     TRISD = 0x80; //RD0:RD6 son salidas
13 }
14
15 unsigned char invertida(unsigned char dato){
16     unsigned char salida = 0;
17     salida = ((dato << 3) & 0x38) + ((dato >> 3) & 0x07) + (dato & 0x40);
18     return salida;
19 }
20
21 void main(void) {
22     configuro();
23     while(1){
24         if (PORTCbits.RC0 ==1){
25             LATD = invertida(holaupc[cuenta]);
26         }
27         else{
28             LATD = holaupc[cuenta];
29         }
30         __delay_ms(300);
31         if (cuenta == 8){
32             cuenta = 0;
33         }
34         else{
35             cuenta++;
36         }
37     }
38 }
```

9

## Ejemplo: Visualizar “HOLA UPC”

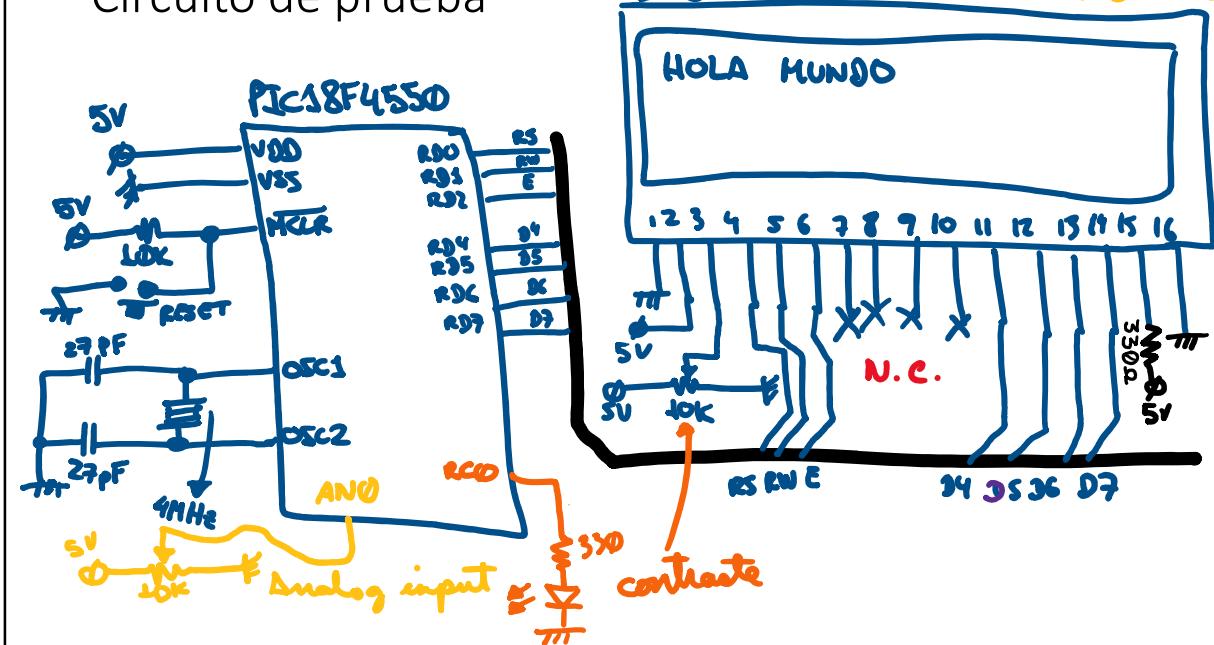
- Código final  
(empleando FOR)

```

1  #include <xc.h>
2  #include "cabecera.h"
3  #define _XTAL_FREQ 48000000UL
4
5  unsigned char msg_hola[] = {0x76, 0x3F, 0x38, 0x77, 0x00, 0x3E, 0x73, 0x39, 0x00};
6
7  void configuro(void){
8     TRISD = 0x80; //RD0:RD6 sobn salidas
9 }
10
11 unsigned char de_cabeza(unsigned char dato){
12     unsigned char salida = 0;
13     salida = ((dato << 3) & 0x38) + ((dato >> 3) & 0x07) + (dato & 0x40);
14     return salida;
15 }
16
17 void main(void) {
18     configuro();
19     while(1){
20         unsigned char x = 0;
21         for(x=0;x<9;x++){
22             if (PORTCbits.RC0 == 1){
23                 LATD = de_cabeza(msg_hola[x]);
24             }
25             else{
26                 LATD = msg_hola[x];
27             }
28             __delay_ms(300);
29         }
30     }
31 }
```

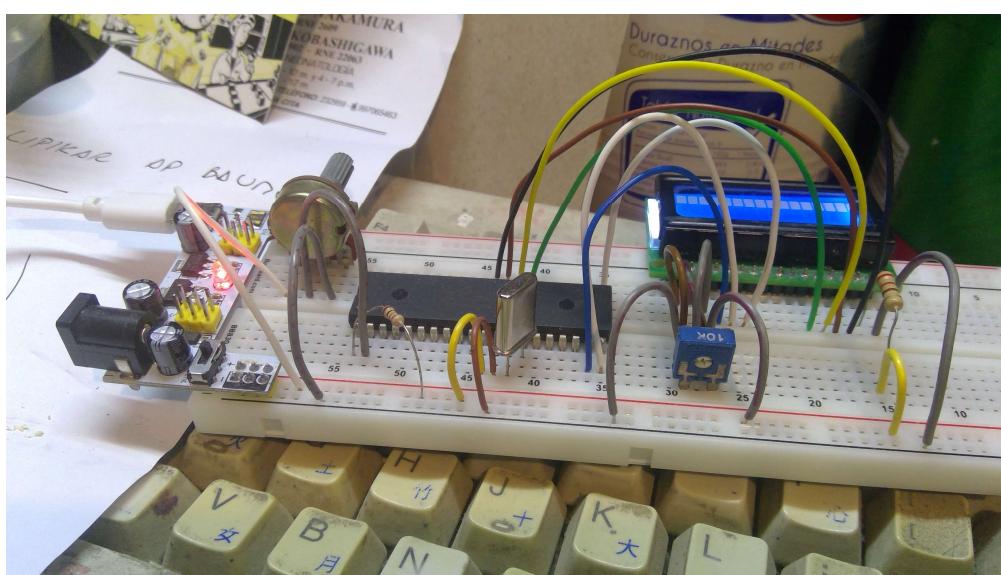
10

## Circuito de prueba



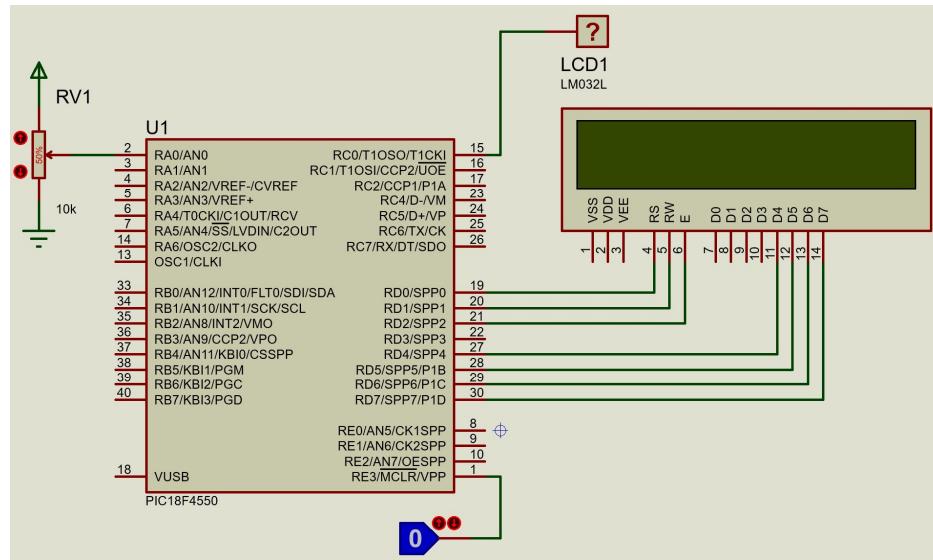
11

## Implementación física



12

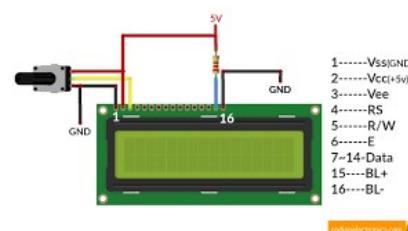
## Circuito simulado en Proteus



13

## El LCD alfanumérico HD44780

- Basado en el controlador Hitachi HD44780A
- Diferentes tamaños, desde 1x8 hasta 4x40
- Interface paralela de datos (4 ó 8 bits)
- Tiene control de contraste y luz de fondo
- Posee un ROM de caracteres predefinidos



14

## El LCD alfanumérico HD44780

- ROM de caracteres:
  - Muy similar al código ASCII en 7 bits
  - El símbolo de grado (°) en ASCII es Alt+0167, en el ROM de caracteres del HD44780 es 0xDF
  - El símbolo “ñ” en ASCII es Alt+164, en el ROM de caracteres del HD44780 es 0xEE
  - Capacidad de ocho caracteres personalizados (CGRAM 0x00-0x07)

Lower R/W Upper R/W Line A/B	0000	0001	0010	0011	0100	0101	0110	0111	0000	0001	0010	0011	0100	0101	0110	0111	
	C5	C6	(1)														
xxxx0000									0 0 P ^ P				- ♪ ミ & p				
xxxx0001	(2)								! 1 A Q a a				・ ♪ フ ゲ & q				
xxxx0010	(3)								" 2 B R b r				「 イ ツ × β θ				
xxxx0011	(4)								# 3 C S c s				」 ウ テ モ ε *				
xxxx0100	(5)								\$ 4 D T d t				、 イ ハ ム φ				
xxxx0101	(6)								% 5 E U e u				・ オ ナ ュ σ Ü				
xxxx0110	(7)								& 6 F V f v				ヲ カ ニ ヨ ρ Σ				
xxxx0111	(8)								' 7 G W g w				ア キ ヴ ラ グ π				
xxxx1000	(1)								( 8 H X h x				イ ク ネ リ ッ ×				
xxxx1001	(2)								) 9 I Y i y				カ テ ル " ピ				
xxxx1010	(3)								* : J Z j z				エ コ ハ レ j ♀				
xxxx1011	(4)								+ ; K [ k {				オ サ ヒ ロ × ハ				
xxxx1100	(5)								, < L ¥ 1				カ シ フ ワ ♀ ヘ				
xxxx1101	(6)								- = M ] m }				ユ ス ベ ハ ナ				
xxxx1110	(7)								. > N ^ n >				ヨ エ ハ ハ ハ				
xxxx1111	(8)								/ ? O _ o <				ツ ツ マ ♀ ö				

15

## El LCD alfanumérico HD44780

- Tabla de caracteres ASCII de 7 bits

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0 000	NUL	(null)	32	20 040	4#32;	Space		64	40 100	4#64;	Ø		96	60 140	4#96;	`	
1	1 001	SOH	(start of heading)	33	21 041	4#33;	!	!	65	41 101	4#65;	A		97	61 141	4#97;	a	
2	2 002	STX	(start of text)	34	22 042	4#34;	"	"	66	42 102	4#66;	B		98	62 142	4#98;	b	
3	3 003	ETX	(end of text)	35	23 043	4#35;	#	#	67	43 103	4#67;	C		99	63 143	4#99;	c	
4	4 004	EOT	(end of transmission)	36	24 044	4#36;	\$	\$	68	44 104	4#68;	D		100	64 144	4#100;	d	
5	5 005	ENQ	(enquiry)	37	25 045	4#37;	%	%	69	45 105	4#69;	E		101	65 145	4#101;	e	
6	6 006	ACK	(acknowledge)	38	26 046	4#38;	&	&	70	46 106	4#70;	F		102	66 146	4#102;	f	
7	7 007	BEL	(bell)	39	27 047	4#39;	:	:	71	47 107	4#71;	G		103	67 147	4#103;	g	
8	8 010	BS	(backspace)	40	28 050	4#40;	{	{	72	48 110	4#72;	H		104	68 150	4#104;	h	
9	9 011	TAB	(horizontal tab)	41	29 051	4#41;	)	)	73	49 111	4#73;	I		105	69 151	4#105;	i	
10	A 012	LF	(NL line feed, new line)	42	2A 052	4#42;	*	*	74	4A 112	4#74;	J		106	6A 152	4#106;	j	
11	B 013	VT	(vertical tab)	43	2B 053	4#43;	+	+	75	4B 113	4#75;	K		107	6B 153	4#107;	k	
12	C 014	FF	(NP form feed, new page)	44	2C 054	4#44;	,	,	76	4C 114	4#76;	L		108	6C 154	4#108;	l	
13	D 015	CR	(carriage return)	45	2D 055	4#45;	-	-	77	4D 115	4#77;	M		109	6D 155	4#109;	m	
14	E 016	SO	(shift out)	46	2E 056	4#46;	.	.	78	4E 116	4#78;	N		110	6E 156	4#10;	n	
15	F 017	SI	(shift in)	47	2F 057	4#47;	/	/	79	4F 117	4#79;	O		111	6F 157	4#111;	o	
16	10 020	DLE	(data link escape)	48	30 060	4#48;	0	0	80	50 120	4#80;	P		112	70 160	4#112;	p	
17	11 021	DC1	(device control 1)	49	31 061	4#49;	1	1	81	51 121	4#81;	Q		113	71 161	4#113;	q	
18	12 022	DC2	(device control 2)	50	32 062	4#50;	2	2	82	52 122	4#82;	R		114	72 162	4#114;	r	
19	13 023	DC3	(device control 3)	51	33 063	4#51;	3	3	83	53 123	4#83;	S		115	73 163	4#115;	s	
20	14 024	DC4	(device control 4)	52	34 064	4#52;	4	4	84	54 124	4#84;	T		116	74 164	4#116;	t	
21	15 025	NAK	(negative acknowledgement)	53	35 065	4#53;	5	5	85	55 125	4#85;	U		117	75 165	4#117;	u	
22	16 026	SYN	(synchronous idle)	54	36 066	4#54;	6	6	86	56 126	4#86;	V		118	76 166	4#118;	v	
23	17 027	ETB	(end of trans. block)	55	37 067	4#55;	7	7	87	57 127	4#87;	W		119	77 167	4#119;	w	
24	18 030	CAN	(cancel)	56	38 070	4#56;	8	8	88	58 130	4#88;	X		120	78 170	4#120;	x	
25	19 031	EM	(end of medium)	57	39 071	4#57;	9	9	89	59 131	4#89;	Y		121	79 171	4#121;	y	
26	1A 032	SUB	(substitute)	58	3A 072	4#58;	:	:	90	5A 132	4#90;	Z		122	7A 172	4#122;	z	
27	1B 033	ESC	(escape)	59	3B 073	4#59;	:	:	91	5B 133	4#91;	[		123	7B 173	4#123;	{	
28	1C 034	FS	(file separator)	60	3C 074	4#60;	<	<	92	5C 134	4#92;	\		124	7C 174	4#124;		
29	1D 035	GS	(group separator)	61	3D 075	4#61;	=	=	93	5D 135	4#93;	]		125	7D 175	4#125;	}	
30	1E 036	RS	(record separator)	62	3E 076	4#62;	>	>	94	5E 136	4#94;	^		126	7E 176	4#126;	DEL	
31	1F 037	US	(unit separator)	63	3F 077	4#63;	?	?	95	5F 137	4#95;	_		127	7F 177	4#127;	DEL	

Source: [www.LookupTables.com](http://www.LookupTables.com)

16

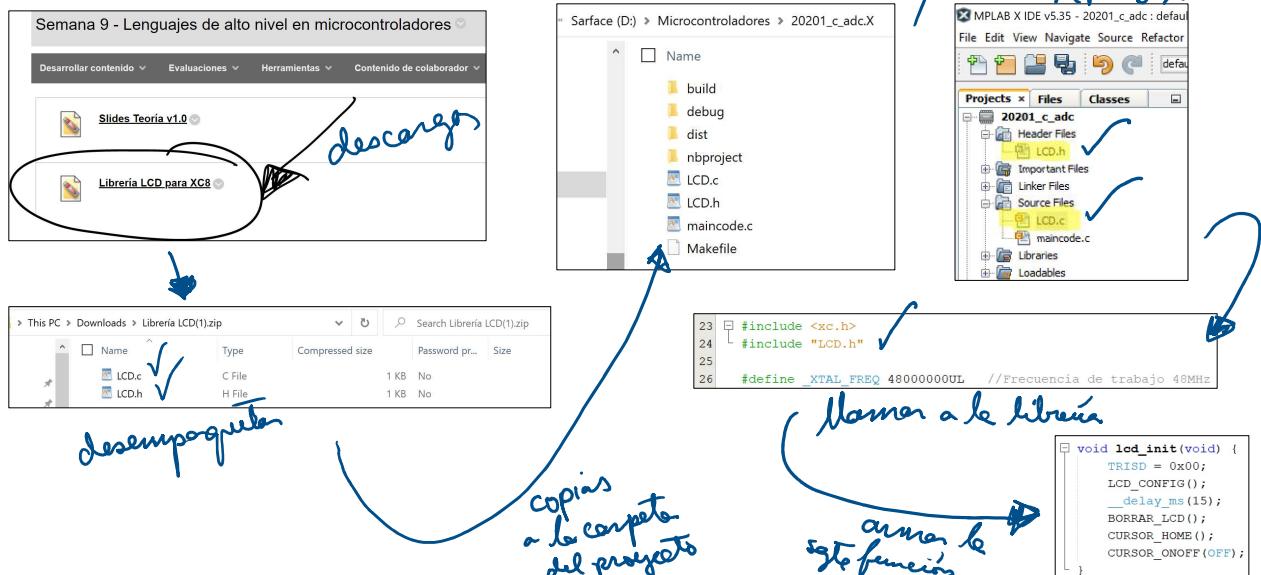
## El LCD alfanumérico HD44780

- Referencia: Hoja técnica del HD44780
  - [http://academy.cba.mit.edu/classes/output\\_devices/44780.pdf](http://academy.cba.mit.edu/classes/output_devices/44780.pdf)
- Para trabajar con el display se ha creado una librería de comandos (desarrollado por el profesor Sergio Salas) en la cual posee las siguientes características:
  - Interface de 4 bits
  - Comandos para: Limpiar pantalla, ocultar cursor, pasar de línea, caracteres personalizados, etc.
  - Puerto D empleado
  - Tener en cuenta FOSC especificado dentro de la librería (48MHz)

17

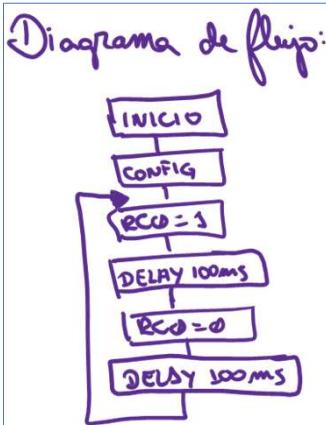
## Uso del LCD en XC8 (librería S\_SAL)

- Crear primero el Proyecto en el MPLABX



18

## Primer ejemplo en XC8: titilador de un bit por RCO



```

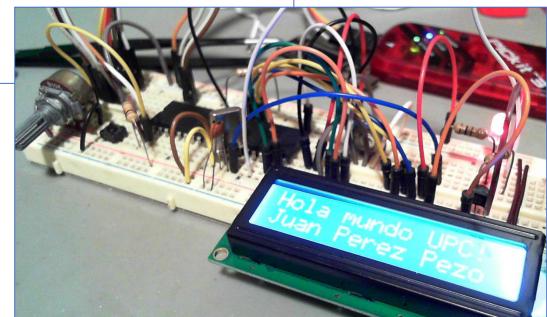
4 #pragma config PLLDIV = 1           // PLL Prescaler Selection
5 #pragma config CPUDIV = OSC1_PLL2 // System Clock Post
6 #pragma config FOSC = XTPLL_XT   // Oscillator Selection
7 #pragma config PWRT = ON          // Power-up Timer Enable
8 #pragma config BOR = OFF          // Brown-out Reset Enable
9 #pragma config WDT = OFF          // Watchdog Timer Enable
10 #pragma config CCP2MX = ON         // CCP2 MUX bit (CCP2)
11 #pragma config PBADEN = OFF        // PORTB A/D Enable bit
12 #pragma config MCLRE = ON          // MCLR Pin Enable bit
13 #pragma config LVP = OFF          // Single-Supply ICSP
14
15 #include <xc.h>
16 #define _XTAL_FREQ 48000000UL
17 /*El microcontrolador tiene funcionando el PLL con 48MHz*/
18
19 void main(void) {
20     TRISCbts.RC0 = 0;           //RC0 salida
21     while(1){
22         LATCbits.LC0 = 1;        //Encendemos el LED
23         __delay_ms(100);        //retardo de 100ms
24         LATCbits.LC0 = 0;        //Apagamos el LED
25         __delay_ms(100);        //retardo de 100ms
26     }
27     return;
28 }
  
```

19

## Segundo ejemplo: Visualizar "Hola mundo UPC!" en el LCD

```

1 //Zona de bits de configuracion
2 #pragma config PLLDIV = 1           // PLL Prescaler Selection
3 #pragma config CPUDIV = OSC1_PLL2 // System Clock Post
4 #pragma config FOSC = XTPLL_XT   // Oscillator Selection
5 #pragma config PWRT = ON          // Power-up Timer Enable
6 #pragma config BOR = OFF          // Brown-out Reset Enable
7 #pragma config BORV = 3            // Brown-out Voltage Selection
8 #pragma config WDT = OFF          // Watchdog Timer Enable
9 #pragma config WDTPS = 32768       // Watchdog Timer Period Selection
10 #pragma config CCP2MX = ON         // CCP2 MUX bit (CCP2)
11 #pragma config PBADEN = OFF        // PORTB A/D Enable bit
12 #pragma config MCLRE = ON          // MCLR Pin Enable bit
13 #pragma config LVP = OFF          // Single-Supply ICSP
14
15 #include <xc.h>
16 #include "LCD.h"
17 #define _XTAL_FREQ 48000000UL
18
19 void init_conf(void){
20     TRISCbts.RC0 = 0;           //RC0 como salida
21     TRISCbts.RC2 = 0;           //RC2 como salida
22 }
23
24 void lcd_config(void){
25     TRISD = 0x00;
26     LCD_CONFIG();
27     __delay_ms(15);
28     Borrar_LCD();
29     Cursor_HOME();
30     Cursor_ONOFF(OFF);
31 }
  
```



20

## Código ejemplo para configurar y leer AN0 en el A/D:

```

28     unsigned int res_ad = 0;
29
30     void configuracion(void) {
31         //Aqui colocas las configuraciones iniciales
32         ADCON2 = 0xA4;           //ADFM=0 (just derecha), 8TAD, Fosc/4
33         ADCON1 = 0x0E;          //Canal AN0 habilitado
34         ADCON0 = 0x01;          //Canal AN0 seleccionado y encendemos el A/D
35
36     void main(void) {
37         configuracion();
38
39         while (1) {
40             //Tu programa de usuario
41             ADCON0bits.GODONE = 1;           //Inicio una captura de muestra en AN0
42             while(ADCON0bits.GODONE == 1);   //Espero a que termine de convertir
43             res_ad = (ADRESH << 8) + ADRESL; //Grabar el resultado en la variable res_ad
44             convierte(res_ad);            //Sacar los dígitos
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66

```

21

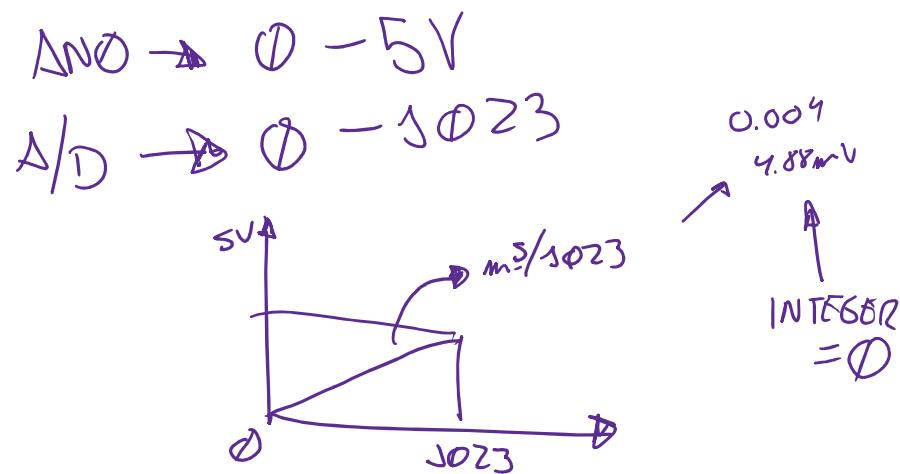
## Código en XC8 del ejemplo inicial:

```

26 #define _XTAL_FREQ 48000000UL //Frecu
27
28 unsigned int res_ad = 0;
29 unsigned int millar = 0;
30 unsigned int centena = 0;
31 unsigned int decena = 0;
32 unsigned int unidad = 0;
33
34 void convierte(unsigned int numero){
35     millar = numero /1000;
36     centena = (numero % 1000) / 100;
37     decena = (numero % 100) / 10;
38     unidad = numero % 10;
39 }
40
41 void lcd_init(void) {
42     TRISD = 0x00;           //Puerto D
43     LCD_CONFIG();
44     __delay_ms(15);
45     BORRAR_LCD();
46     CURSOR_HOME();
47     CURSOR_ONOFF(OFF);
48 }
49
50 void configuracion(void) {
51     //Aqui colocas las configuraciones
52     ADCON2 = 0xA4;          //ADE
53     ADCON1 = 0x0E;          //Canal A
54     ADCON0 = 0x01;          //Canal A
55     lcd_init();
56 }
57
58 void main(void) {
59     configuracion();
60     ESCRIBE_MENSAJE("VIRTUAlASO",10);
61     while (1) {
62         //Tu programa de usuario
63         ADCON0bits.GODONE = 1;
64         while(ADCON0bits.GODONE == 1);
65         res_ad = (ADRESH << 8) + ADRESL;
66         convierte(res_ad);
67         POS_CURSOR(2,0);
68         ENVIA_CHAR(millar+0x30);
69         ENVIA_CHAR(centena+0x30);
70         ENVIA_CHAR(decena+0x30);
71         ENVIA_CHAR(unidad+0x30);
72     }
73 }
```

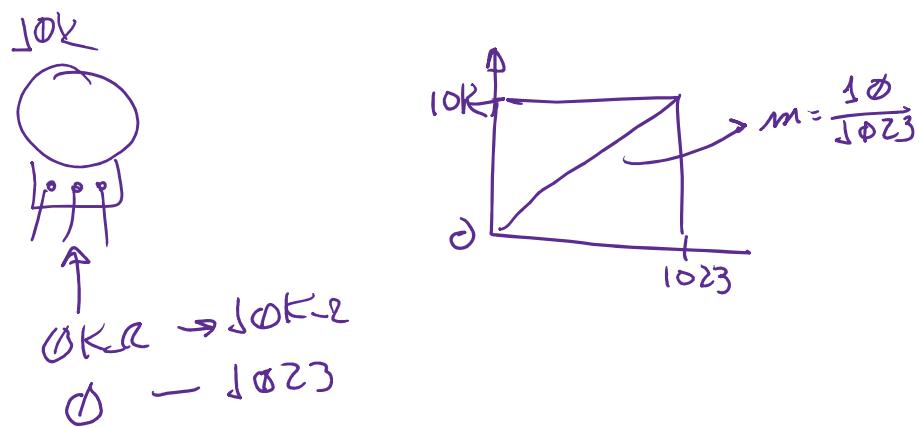
22

## Proceso de escalamiento



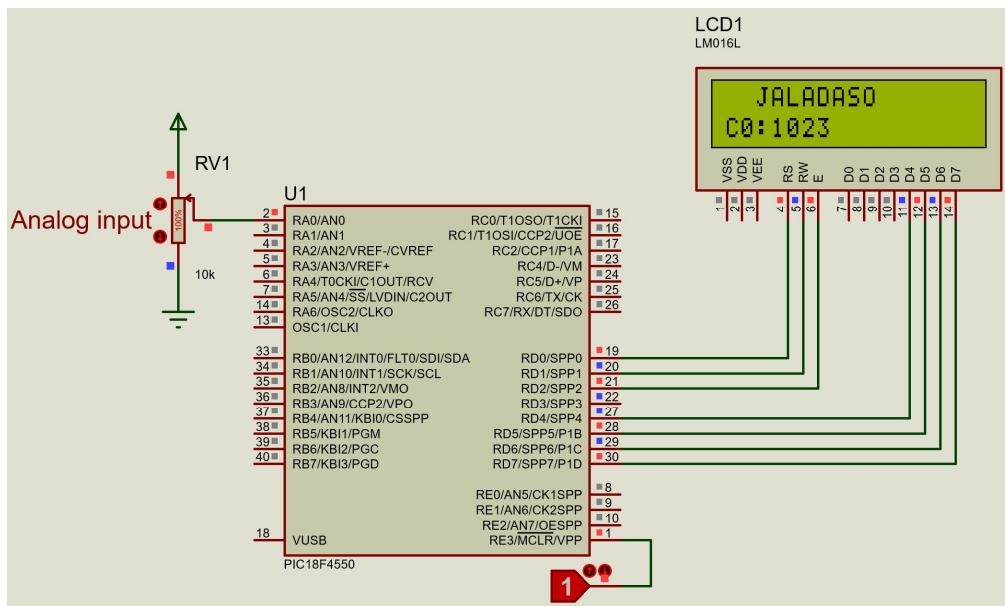
23

## Proceso de escalamiento (valor del pot)



24

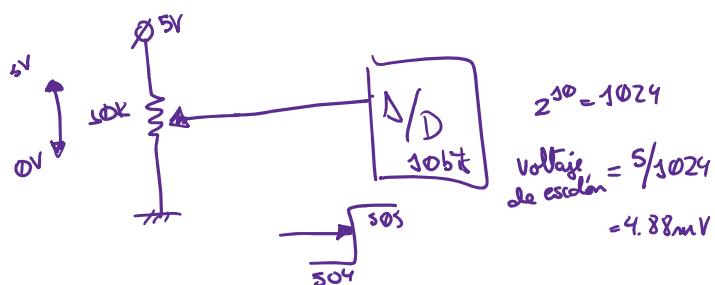
## Simulación en Proteus:



25

## Observaciones:

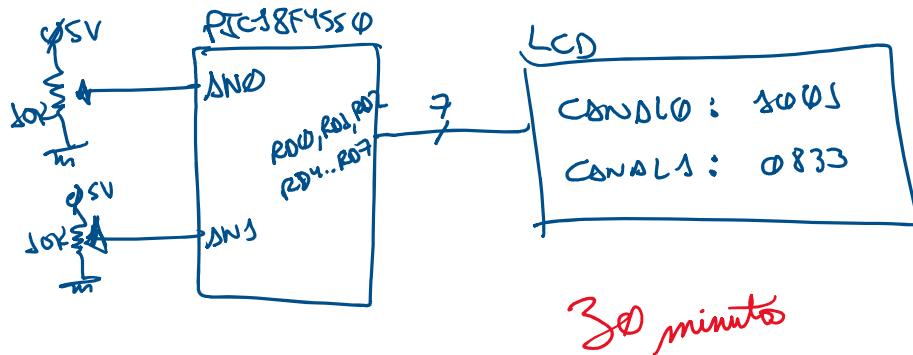
- El último dígito que se muestra (de las cuentas de 10 bits del resultado del A/D) varía bastante:



26

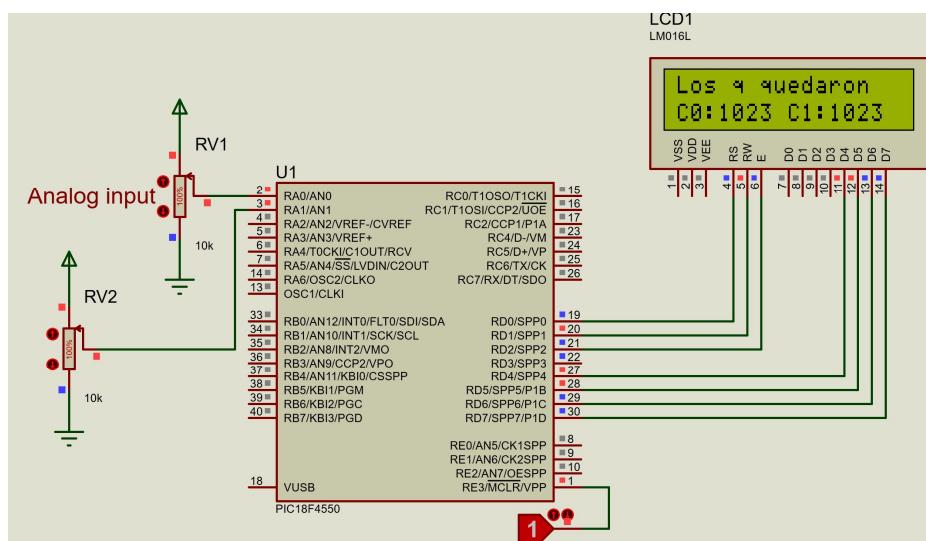
## Asignación:

- Leer dos canales analógicos y mostrarlos en el LCD



27

## Simulación:



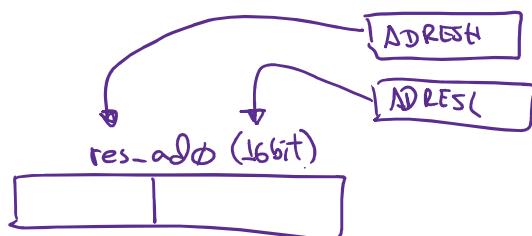
28

## Código en XC8:

```

27  void configuracion(void) {
28      //Aquí colocas las configuraciones iniciales
29      ADCON2 = 0xA4;           //ADFM=0 (Canal AN0)
30      ADCON1 = 0x0D;           //Canal AN0
31      ADCONbits.ADON = 1;     //Encendemos el ADC
32      lcd_init();
33
34      unsigned int res_ad0 = 0;
35      unsigned int res_ad1 = 0;
36
37      unsigned int millar = 0;
38      unsigned int centena = 0;
39      unsigned int decena = 0;
40      unsigned int unidad = 0;
41
42      void convierte(unsigned int numero){
43          millar = numero /1000;
44          centena = (numero % 1000) / 100;
45          decena = (numero % 100) / 10;
46          unidad = numero % 10;
47
48      void lcd_init(void) {
49          TRISD = 0x00;           //Puedo escribir en el LCD
50          LCD_CONFIG();
51          _delay_ms(15);
52          BORRAR_LCD();
53          CURSOR_HOME();
54          CURSOR_ONOFF(OFF);
55
56      void main(void) {
57          configuracion();        //Llamada a la función de configuración
58          ESCRIBE_MENSAJE("Los q quedaron",14);
59          while(1){
60              ADCON0 = 0x03;
61              //ADCON0bits.GODONE = 1;
62              while(ADCON0bits.GODONE == 1);    //Esperamos a que se complete la conversión
63              res_ad0 = (ADRESH << 8) + ADRESL; //Concatenamos los resultados
64              ADCON0 = 0x07;                   //Configuramos para leer el canal AN1
65              while(ADCON0bits.GODONE == 1);    //Esperamos a que se complete la conversión
66              res_ad1 = (ADRESH << 8) + ADRESL; //Concatenamos los resultados
67              POS_CURSOR(2,0);               //Movemos el cursor a la posición (2,0)
68              ESCRIBE_MENSAJE("CO:",3);       //Escribimos "CO:" en la pantalla
69              convierte(res_ad0);           //Convertimos el resultado de la ADC0
70              ENVIA_CHAR(millar+0x30);      //Enviamos el carácter correspondiente
71              ENVIA_CHAR(centena+0x30);     //Enviamos el carácter correspondiente
72              ENVIA_CHAR(decena+0x30);      //Enviamos el carácter correspondiente
73              ENVIA_CHAR(unidad+0x30);      //Enviamos el carácter correspondiente
74              ESCRIBE_MENSAJE(" Cl:",4);    //Escribimos " Cl:" en la pantalla
75              convierte(res_ad1);           //Convertimos el resultado de la ADC1
76              ENVIA_CHAR(millar+0x30);      //Enviamos el carácter correspondiente
77              ENVIA_CHAR(centena+0x30);     //Enviamos el carácter correspondiente
78              ENVIA_CHAR(decena+0x30);      //Enviamos el carácter correspondiente
79              ENVIA_CHAR(unidad+0x30);      //Enviamos el carácter correspondiente
80
81
82
83
84
85
86
87
88      }
  
```

29



30

Fin de la sesión!