Práctica 5

Comunicación serie y entradas analógicas

1. Objetivos

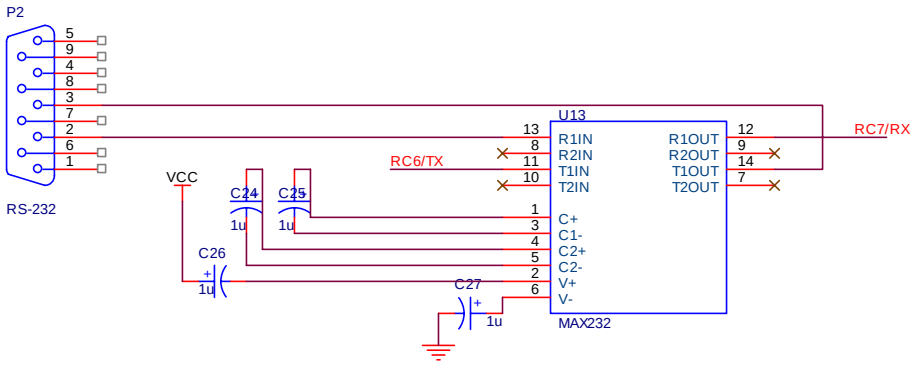
El objetivo de esta sesión es que el alumno aprenda a programar la EUART interna del PIC, de esta manera puede comunicar datos en serie a un PC, por ejemplo. Como aplicación complementaria se hará uso de los conversores internos Analógico-Digitales (A/D) y las fuentes analógicas incorporadas en la tarjeta EduMIC.

2. Introducción

Los elementos de la tarjeta que se introducen en esta sesión son el puerto de comunicaciones serie RS232 y las cuatro fuentes analógicas integradas: sensor de luminosidad LDR, sensor de temperatura y dos potenciómetros ajustables.

2.1 Puerto serie de comunicaciones RS232

Muchos de los PIC poseen internamente una EUART que se puede programar para la transmisión/recepción de datos en serie de forma asíncrona. Este es precisamente el tipo de comunicación serie que se utiliza en el interfaz RS232. En la siguiente figura se representa el circuito externo al PIC necesario para adaptar las señales TTL de la EUART del PIC a las propias de las especificaciones del RS232:



El integrado MAX232 (ST232) se encarga de traducir las señales de 5 voltios del PIC en las señales de 12 voltios que se manejan en el interfaz RS232. Las únicas señales necesarias para la conexión son los bits 6 y 7 del puerto C del PIC.

La programación del interfaz serie del PIC es muy sencilla, basta con poner los valores adecuados en los registros `SPBRG`, `TXSTA`, `RCSTA` y `BAUDCON`, además hay que programar el pin `RC6` como salida y el `RC7` como entrada. Para ello `TRISC` debe ser:

```
TRISC= 0b10000000; // RC6 salida, RC7 entrada, resto nada
```

Para generar la velocidad de transmisión existe un temporizador dedicado `BRG`, que se incrementa por impulsos de la señal de reloj interna Fosc. Puede funcionar en modo 8 bits (bit `BRG16=0`) o 16 bits (bit `BRG16=1`). El bit `BRG16` está en el registro `BAUDCON` y se puede acceder a él mediante la estructura `BAUDCONbits.BRG16`.

El valor de cuenta se almacena en `SPBRH:SPBRG`. Existen un predivisor por 4, 16 ó 64, que se selecciona mediante los bits `BRG16` y `BRGH`. La velocidad en baudios se controla mediante el registro `SPBRH:SPBRG` y las siguientes fórmulas:

TABLA: BAUDIOS

Bits			BRG/EUSART Modo	Baudios
SYNC	BRG16	BRGH		
0	0	0	8-bit/Asincrono	Fosc/[64 (n + 1)]
0	0	1	8-bit/Asincrono	Fosc/[16 (n + 1)]
0	1	0	16-bit/Asincrono	
0	1	1	16-bit/Asincrono	Fosc/[4 (n + 1)]
1	0	x	8-bit/Sincrono	
1	1	x	16-bit/Sincrono	

Por ejemplo, para una velocidad de 9600 baudios con receptor en modo asíncrono (**SYNC=0**) con alta velocidad (**BRGH=1**) y recarga de 8 bits solamente (**BRG16=0**), la fórmula para el cálculo del valor de recarga de SPBRG es **Fosc/[16(SPBRG +1)]**. Despejando el valor de SPBRG para Fosc=4 MHz (frecuencia del oscilador en las tarjetas EduMIC) SPBRG=25. El resumen de los registros asociados con el cálculo del valor de recarga es:

TABLA: REGISTROS ASOCIADOS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	53
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	53
BAUDCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	53
SPBRGH	EUSART Baud Rate Generator Register High Byte								53
SPBRG	EUSART Baud Rate Generator Register Low Byte								53

Además hay que programar el funcionamiento del puerto como: asíncrono, de 8 bits, con recepción y transmisión continua. Estas características se programan sobre los registros TXSTA y RCSTA:

REGISTRO: TXSTA

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
bit 7							bit 0

bit 7 **CSRC: Selección de la fuente de reloj**
Asincrono modo:
No importa.
Sincrono mode:
1 = Master modo
0 = Slave modo

bit 6 **TX9:** Transmisión de 9-bits
1 = Selecciona 9-bits
0 = Selecciona 8-bits

bit 5 **TXEN:** Bit de habilitación de la transmisión
1 = Habilita la transmisión
0 = Deshabilita la transmisión

bit 4 **SYNC:** Modo de la EUSART
1 = Sincrono
0 = Asíncrono

bit 3 **SENDB:** Enviar caracter de Break
Asíncrono modo:
1 =Enviar caracter de Break
0 =Se ha completado el envío
Síncrono modo:
No importa.

bit 2 **BRGH:** Selección de alta frecuencia
Asíncrono modo:
1 =Alta velocidad
0 =Baja velocidad
Síncrono modo:
No se usa.

bit 1 **TRMT:** Bit de habilitación de la transmisión
1 = TSR vacío
0 = TSR lleno

bit 0 **TX9D: 9th** bit de datos

REGISTRO: RCSTA

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

bit 7 **SPEN: Bit de habilitación de la EUSART**
1 = Puerto serie habilitado
0 = Deshabilitado

bit 6 **RX9:** Se habilita recepción del 9-bit
1 = Selecciona 9-bit
0 = Selecciona 8-bit

bit 5 **SREN:** Habilita la recepción de un sólo dato
Asíncrono:
No se utiliza.
Síncrono modo – Master:
1 = Habilita la recepción de un sólo dato
0 =Se ha completado la recepción
.
Síncrono modo – Slave:
No importa.

bit 4 **CREN:** Bit de habilitación de la recepción
Asíncrono:
1 =Habilita
0 =Deshabilita
Síncrono:
1 =Habilita
0 =Deshabilita

bit 3 **ADDEN:** Detección de la dirección
Asíncrono con 9-bit (RX9 = 1):
1 =Habilita la detección del noveno bit a '1', sólo si el dato tiene el noveno bit a '1' lo recibe

0 =Deshabilita la detección del noveno bit a '1', recibe todos los datos.

bit 2 **FERR:** Error de encuadre
1 =Error de encuadre, bit de stop es un '0'.
0 = No framing error

bit 1 **OERR:** Error de sobre escritura
1 =Error de sobreescritura
0 =No hay error

bit 0 **RX9D:** Noveno bit de los datos recibidos

Los valores de estos dos registros deben ser:

```
TXSTA=0b00100100;     // Asincrono, alta velocidad, 8 bits, transmisión hab.
RCSTA=0b10010000;    // Habilita el puerto y la recep de 8 bits,
```

Una vez iniciado el interfaz serie ya se pueden enviar y recibir datos de 8 bits. El envío de un byte se realiza simplemente escribiendo en el registro `TXREG` y la lectura se realiza leyendo el registro `RCREG`.

Es muy importante sincronizar correctamente estas señales, es decir, si se quieren enviar varios caracteres seguidos no se pueden escribir en `TXREG` a la velocidad del procesador, ya que entonces se perderían la mayoría de los datos (el procesador es mucho más rápido que la transmisión serie). Para estar seguros de que el último byte que se escribió en TXREG se ha pasado al registro de desplazamiento TSR, hay que mirar el bit 4 (`TXIF`) del registro `PIR1` para ver si el buffer de transmisión está lleno o vacío; si esta lleno (valor 0) no podemos escribir un nuevo dato en TXREG, si está vació (valor 1) si podemos. Este mismo bit sirve para que un manejador de interrupción sepa qué periférico produjo la interrupción, ya que cada vez que se vacía el buffer TXREG se produce una interrupción (para que la rutina de tratamiento de la interrupción pueda enviar un nuevo byte).

Con la recepción pasa algo parecido. Cuando se recibe algo por el puerto serie (en `RCREG`) se produce una interrupción y se activa el bit 5 (`RCIF`) del registro `PIR1`, indicando que hay algo en `RCREG` para leer.

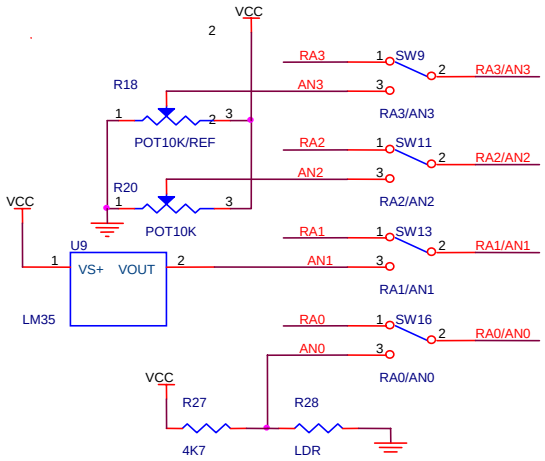
Las interrupciones del puerto serie se activan con los bits 4 (`TXIE`) y 5 (`RCIE`) del registro `PIE1` además de los bits `PEIE` y `GIE` del `INTCON` (habilitación de las interrupciones en general y de periféricos en particular). El resumen de los registros asociados es:

TABLA: REGISTROS ASOCIADOS

Nombre	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	SPPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
PIE1	SPPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
IPR1	SPPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
TXREG	EUSART Transmit Register							
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
BAUDCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN
SPBRGH	EUSART Baud Rate Generator Register High Byte							
SPBRG	EUSART Baud Rate Generator Register Low Byte							

2.2 Las entradas analógicas

En la tarjeta EduMIC se han incluido 4 fuentes de tensión conectadas a 4 de los conversores A/D del PIC. El circuito correspondiente a estas fuentes (sensores) se muestra en la siguiente figura:



El sensor de luminosidad: se trata de una resistencia LDR cuyo valor óhmico varía según la intensidad de la luz incidente. Se ha puesto en serie con una resistencia de 4K7 formando un divisor de tensión. Esta fuente está conectada al pin `AN0` (bit 0 del puerto A) del PIC.

El sensor de temperatura: se trata de un integrado (LM35) cuya tensión de salida es directamente proporcional a la temperatura a razón de 10 mV/ºC, siendo la salida de 0 voltios a 0 ºC. Este dispositivo está conectado a la entrada `AN1` del puerto A.

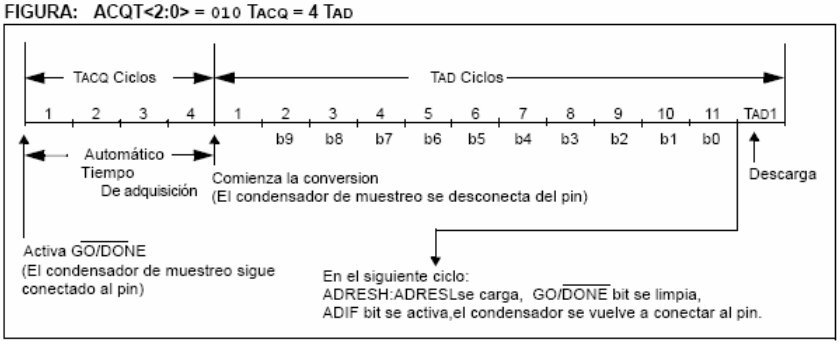
Fuentes lineales de tensión: Se proveen también dos potenciómetros que permiten ajustar cualquier tensión entre 0 y 5 voltios. Las fuentes están conectadas a los puertos `AN2` y `AN3`.

En la figura anterior se muestran también los conmutadores que permiten elegir entre entradas digitales (`RA`), analógicas (`AN`) o desconectadas. **Es evidente que para esta sesión las entradas al PORTA deben estar en posición analógica.**

Los registros implicados en la preparación de una conversión analógica-digital son el `ADCON0, ADCON1` y `ADCON2`. El primero sirve para configurar, el canal de entrada, activación de la conversión y encendido del módulo A/D. El segundo registro sirve para configurar las entradas del puerto A como mezcla entre digitales y analógicas. El tercero sirve para especificar el formato (justificación) del valor devuelto y el reloj de la conversión.

La conversión empieza por ajustar los pines del puerto A como entradas. Luego se programan los registros `ADCON0, ADCON1` y `ACON2` con los valores seleccionados. La conversión está formada por dos periodos: Adquisición (TACQ) y Conversión (TAD):

FIGURA: ACQT<2:0> = 010 TACQ = 4 TAD



La programación de ambos periodos se hace mediante los bits `ACQT2:ACQT0` y `ADCS2:ADCS0` del registro `ADCON2`. Por ejemplo para Fosc=4Mhz como `TAD` debe estar en el rango (0,7-25)μ el valor de los bits `ADCS2:ADCS0` deben ser por lo menos:

`TAD=(1/Fosc)*divisor => (0,7-25)μ = (1/4Mhz)*divisor => divisor=4;`

Por lo tanto `TAD= (4/4Mhz)= 1μ =>` **`ADCS2:ADCS0=100.`**

Además **`ACQT2:ACQT0=111,`** `=> TACQ=20TAD=20*1μ.`

REGISTRO: ADCON0: A/D CONTROL REGISTER 0

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON
bit 7							bit 0

bit 7-6 **No implementado:** Se lee como '0'

bit 5-2 **CHS3:CHS0:** Selección del canal analógico
0000 = Canal 0 (AN0)
0001 = Canal 1 (AN1)
0010 = Canal 2 (AN2)
0011 = Canal 3 (AN3)
0100 = Canal 4 (AN4)
0101 = Canal 5 (AN5)(1,2)
0110 = Canal 6 (AN6)(1,2)
0111 = Canal 7 (AN7)(1,2)
1000 = Canal 8 (AN8)
1001 = Canal 9 (AN9)
1010 = Canal 10 (AN10)
1011 = Canal 11 (AN11)
1100 = Canal 12 (AN12
1101 = No implementado(2)
1110 = No implementado(2)
1111 = No implementado(2)

Note 1: No están implementados en los dispositivos de 28-pines.

bit 1 **GO/DONE:** Inicio y estado de la conversión
SI ADON = 1:
1 = A/D conversión en progreso
0 = A/D parado

bit 0 **ADON:** A/D Bit de encendido del módulo A/D
1 = A/D el módulo está habilitado
0 = A/D el módulo no está habilitado

REGISTRO: **ADCON1: A/D CONTROL REGISTER 1**

U-0	U-0	R/W-0	R/W-0	R/W-0(1)	R/W(1)	R/W(1)	R/W(1)
—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7-6 **No implementado:** Se lee como '0'

bit 5 **VCFG1:** Configuración del voltaje de referencia (V_{REF-})
1 = V_{REF-} (AN2)
0 = V_{SS}

bit 4 **VCFG0:** Configuración del voltaje de referencia (V_{REF+})
1 = V_{REF+} (AN3)
0 = V_{DD}

bit 3-0 **PCFG3:PCFG0:** A/D Bits de configuración de las entradas:

PCFG3: PCFG0	AN12	AN11	AN10	AN9	AN8	AN7(2)	AN6(2)	AN5(2)	AN4	AN3	AN2	AN1	AN0
0000(1)	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	A	A	A	A	A	A	A	A	A
0111(1)	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D

A = Analógico D = Digital

Note 1: El valor del PCFG despues de un reset depende de PBADEN Si PBADEN = 1, PCFG<3:0> = 0000; si PBADEN = 0, PCFG<3:0> = 0111.
2: Solo en dispositivos de 40/44-pin.

REGISTRO: **ADCON2: A/D CONTROL REGISTER 2**

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
bit 7							bit 0

bit 7 **ADFM:** A/D Ajusta el formato del resultado
1 = Justificación derecha
0 = Justificación izquierda

bit 6 **No implementado:** Se lee como '0'

bit 5-3 **ACQT2:ACQT0:** A/D Selección del tiempo de adquisición
111 = 20 T_{AD}
110 = 16 T_{AD}
101 = 12 T_{AD}
100 = 8 T_{AD}
011 = 6 T_{AD}
010 = 4 T_{AD}
001 = 2 T_{AD}
000 = 0 T_{AD}(1)

bit 2-0 **ADCS2:ADCS0:** A/D Selección del reloj para el periodo de conversión
111 = F_{RC} (derivado del oscilador RC)
110 = F_{OSC}/64
101 = F_{OSC}/16
100 = F_{OSC}/4
011 = F_{RC} (derivado del oscilador RC)
010 = F_{OSC}/32
001 = F_{OSC}/8
000 = F_{OSC}/2

Luego se debe activar el bit `GO_DONE` que da comienzo a la conversión. A continuación debemos esperar a que acabe la conversión, para ello se monitoriza el propio bit `GO_DONE` hasta que se haga cero (se terminó la conversión). Posteriormente se puede recuperar el resultado de 10 bits en los registros `ADRESH` y `ADRESL` según la alineación (derecha o izquierda) que se hubiera elegido.

El siguiente trozo de código muestra cómo realizar la lectura de un valor analógico siguiendo los pasos explicados anteriormente. En este ejemplo se ha especificado un reloj de conversión Fosc/4, se ha elegido la entrada `AN0`, todos los pins son analógicos y el resultado está alineado a la derecha (8 bits bajos en `ADRESL` y los dos altos en `ADRESH`):

```
/* en la parte de configuración*/
   TRISA=0xFF;            //Puerto A de Entrada
   ADCON1=0x0A;           //Todos los bits de A son analógicos,
   ADCON0=0x01;           //Enciende A/D, conversor parado, canal de entrada AN0
   ADCON2=0b10111100;     //ADFM=1 justificación derecha,
                          //Tiempo de adquisicion ACQT2:ACQT0=111 => 20TAD
                          //Periodo TAD ADCS2:ADS0=100=> TAD=4Tosc=> (1/4Mhz)*4=1µ
/* en la parte de conversión*/
   ADCON0bits.GO_DONE=1;    // Empieza la adquisición
   while (ADCON0bits.GO_DONE);
   valor=ADRESH;
   valor=(valor<<8)+ADRESL;
```

3. Trabajos a realizar

En todas las sesiones de laboratorio se pide la realización de unos programas o tareas. **Es imprescindible que se tengan preparados, o al menos esbozados, los programas al principio de cada sesión**. Al inicio de la sesión el profesor comprueba de que efectivamente los programas han sido preparados y se pone la nota correspondiente. Tener los programas preparados al inicio de la sesión es muy importante, de lo contrario, es muy difícil que en una misma sesión se puedan terminar los programas habiendo probado su funcionamiento.

Programa 1: Comunicación serie

Se propone realizar un programa para el microcontrolador que reenvíe todos los caracteres recibidos por el puerto serie. La velocidad de transmisión será 9600baudios con 8bits sin paridad y la recepción de los caracteres se detectará mediante interrupción. Para ello se deben habilitar las interrupciones de recepción del puerto serie, cuando dicha interrupción se activa se habrá recibido un carácter por el puerto serie del microcontrolador. El mismo carácter leído se reenviará en sentido contrario, de forma que si conectamos el micro con un PC a través de un cable serie veremos el eco de todos los caracteres que se envíen desde el PC al microcontrolador. Para comprobar el funcionamiento del programa en el PC se tiene un software denominado “Comunica” (disponible en la página web de la asignatura) que muestra por pantalla todo lo que se recibe por el puerto serie y envía también por este puerto el ASCII del carácter que se escribe por el teclado del PC.

Recordar que antes de programar el microcontrolador con el software de programación EduMIC, la palabra de configuración config1 hay que modificarla para que su valor sea **config1= ‘0C00’.**

Programa 2: Control remoto de las entradas analógicas del PIC

El objetivo de este programa es poder leer de forma individualizada y remota las cuatro entradas analógicas correspondientes a las cuatro fuentes de la tarjeta EduMIC. Para ello conectaremos la tarjeta al PC a través del puerto serie a 9600 baudios.

El programa leerá de forma secuencial cada segundo una entrada analógica formando un bucle secuencial e infinito, y enviará el valor digital de cada una de las lecturas por el puerto serie como una secuencia de 4 carácteres ASCII hacia el PC. Los caracteres transmitidos se pueden incluir en una cadena que además debe incluir el nº de sensor, la cadena a transmitir sería:

volatile char car[8]={ 's' , '0' , ':' , ' ' , ' ' , ' ' , ' ' , 13}; //13 es el salto de línea

Donde el ‘0’ identifica al sensor y debería actualizarse en cada nuevo envío y los 4 caracteres ‘ ‘ son los espacios reservados para escribir el valor digital del sensor, donde se deben escribir los 4 dígitos como 4 caracteres ASCII.

Para transmitir correctamente la cadena, después de haber actualizado sus valores, es importante no transmitir un nuevo valor hasta que no haya concluido la transmisión del anterior, tal y como se comentaba en la introducción. Esta sincronización se realizará por chequeo continuo del bit `TXIF` del registro `PIR1` en vez de por interrupciones. (El retraso de 1 seg entre lectura y lectura se puede implementar con una subrutina de retraso mediante la función *Delay10KTCYx(unsigned char unit)*)

Es importante que los conmutadores de la placa EduMIC que sirven para elegir entre Analógico o Digital estén en su posición central (Analógico) de esta manera las entradas del puerto A se conectan a los sensores de los cuales se va leer.