

Microcontroladores

Semana 12-13

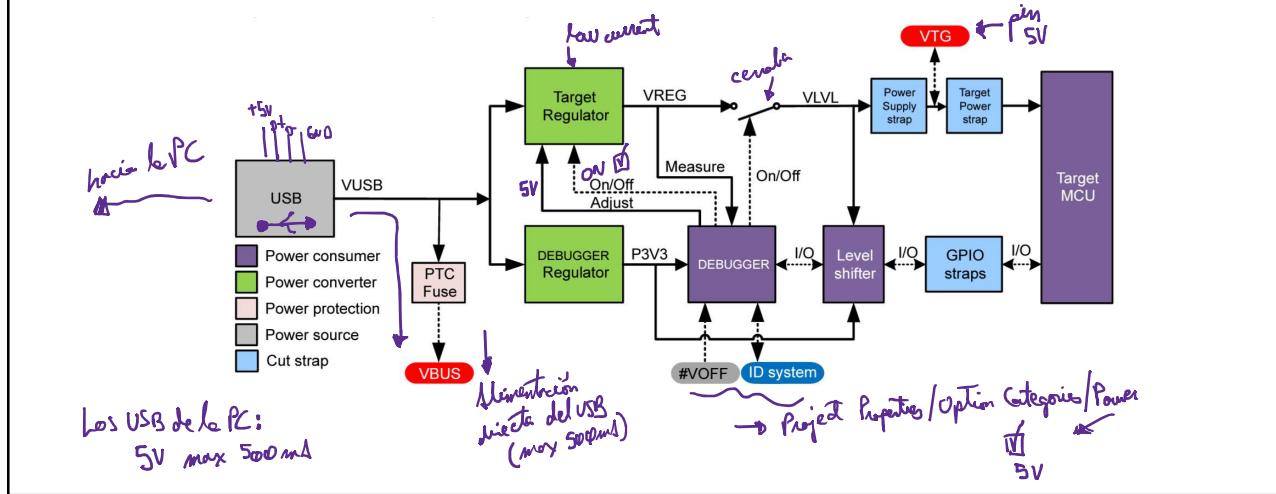
Profesor Kalun Lau

Preguntas previas

- Tuve un problema con errores al momento de programar el curiosity nano, me salía error aun cambiando de VBUS a VTG, abriendo y cerrando la aplicación del MPLABX, reiniciando la PC y conectando y desconectado el cable USB, también intenté cambiar a otro cable y nada, pero hoy arranqué todo y funciona normal. ¿Qué ha pasado?
 - Muy posible problema de firmware del curiosity nano

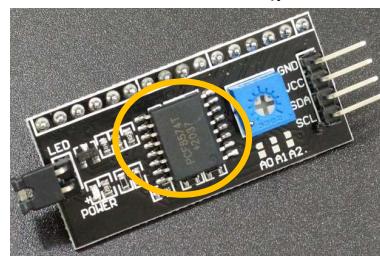
Preguntas previas (cont...)

- No he seteado el Power a 5V en las propiedades del proyecto y aun así prende mi circuito de prueba. ¿A qué se debe?
 - Por defecto el VTG arroja 3.3V si es que no has seteado para que entregue 5V

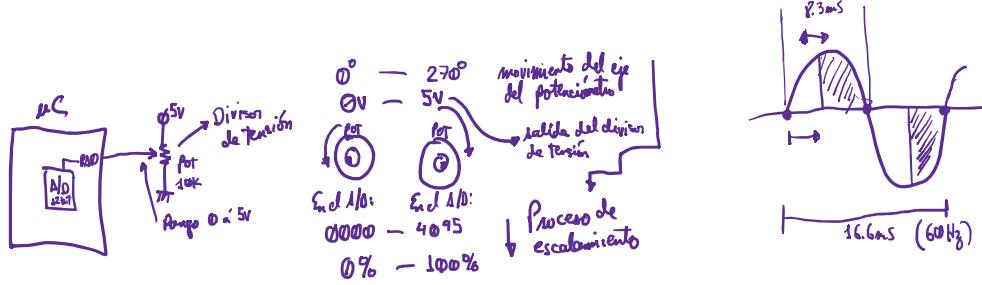


Atención de consultas 2024-2

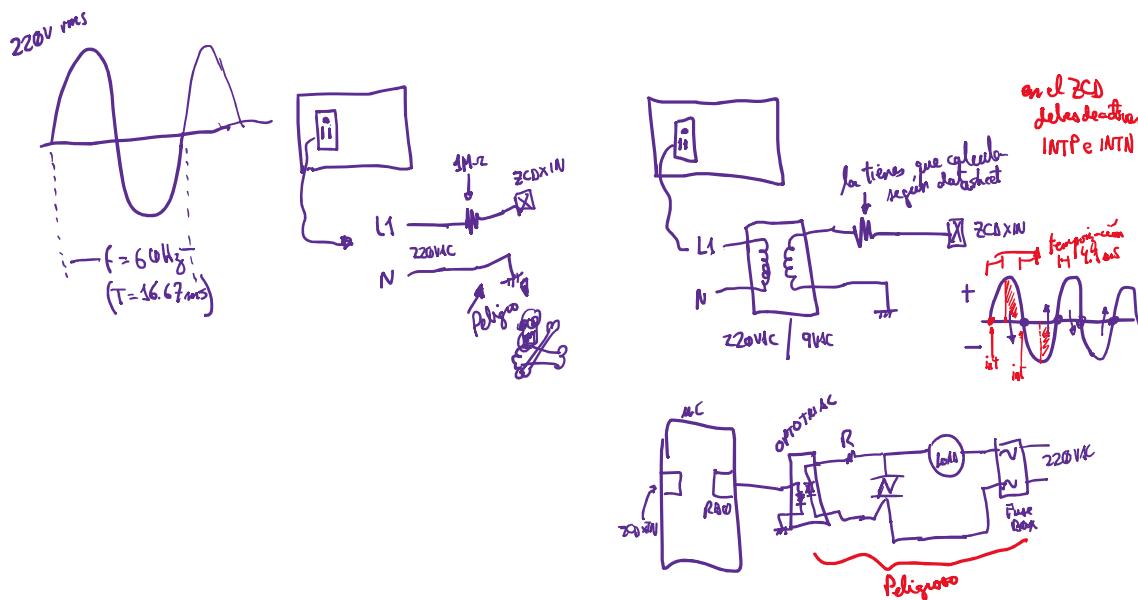
- ¿Vamos a ver comunicación serial en el curso?
 - Si, en esta semana se atiende dicho tema
- ¿Cómo hago para reportar si un integrante de grupo ya no participa en las actividades del curso?
 - Se reporta la incidencia en la hoja de Excel publicada en la sección de indicaciones de TF en el AV.
- He conseguido un I2C-LCD pero no funciona, no inicializa
 - Muy posiblemente el chip de la tarjeta es un PCF8574A en lugar de un PCF8574, solo hay que cambiar la dirección de esclavo de 0x27 (para PCF8574) a 0x3F (para PCF8574A)



Sesión de consultas



Sesión de consultas:



Atección de consultas 2025-1

- ¿El sensor MQ2 tiene interface I2C?
 - No, se tiene que usar una entrada analógica para leer la señal que te arroja dicho sensor



- ¿Se pueden conectar más de un LCD por el I2C?
 - Si, teniendo en cuenta que se debe de cambiar la dirección I2C físicamente en alguno de las dos tarjetas para que se pueda visualizar mensajes diferentes en ambos LCDs.
- El DS18B20 usa 1-wire. ¿Qué es eso?
 - Es un protocolo de comunicación serial desarrollado por Dallas Semiconductor en el cuál emplea un solo cable para la comunicación con el periférico y también por el mismo cable puede energizarlo (activando la opción de Parasite Power)

Preguntas 2025-1

- Puedo usar el ADC con interrupciones?
 - Si, se tiene que habilitar el ADIE (PIE1 bit ADIE), tener en consideración que dentro de la función de atención a esta interrupción deberá de obtener el dato del resultado de la conversión y alojarla en una variable global.
- ¿Se tienen que colocar mas resistencias en las líneas de I2C si es que se usan mas dispositivos en las líneas de I2C?
 - No, solo se colocan dos resistencias: una de pullup en SDA y otra de pullup en SCL, las resistencias de valor 4.7K
- ¿Los displays OLED se usa también I2C?
 - Hay dos interfaces: SPI y I2C para los displays OLEDs
 - Ambos usan el controlador SSD1306
 - SH1106

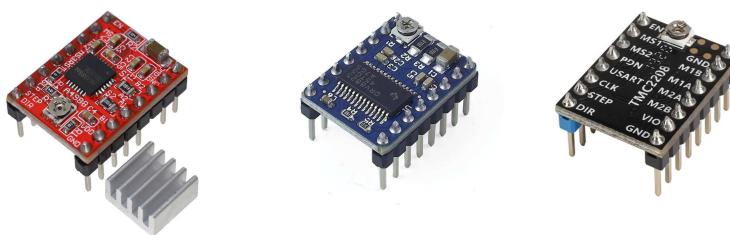


Preguntas 2025-1

- ¿En qué programa se realizan las PCB?
 - Los mas utilizados: Autodesk Eagle, Autodesk Fusion, EasyEDA, KiCAD, Altium Designer.
- ¿Qué componentes electrónicos usaremos en la semana?
 - Servomecanismos, el sensor ultrasónico HC-SR04, el sensor de temperatura y humedad DHT11 y DHT22, Neopixel
- ¿Fabricantes de PCB?
 - Jlcpcb.com, Pcbway.com
- ¿Puede hacernos un ejemplo de diseño de PCB?

Preguntas 2025-2

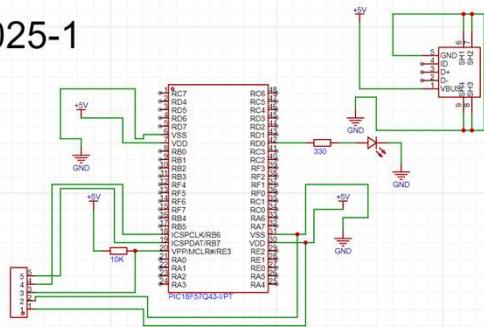
- ¿Cómo puedo mover un motor de pasos, con el Timer?
 - Se tienen comercialmente diferentes módulos driver para motores de pasos:
 - A4988
 - DRV8825
 - TMC2208
- Tienen diferentes maneras de recibir las señales de movimiento desde el microcontrolador:
 - Tren de pulsos
 - I2C ó UART



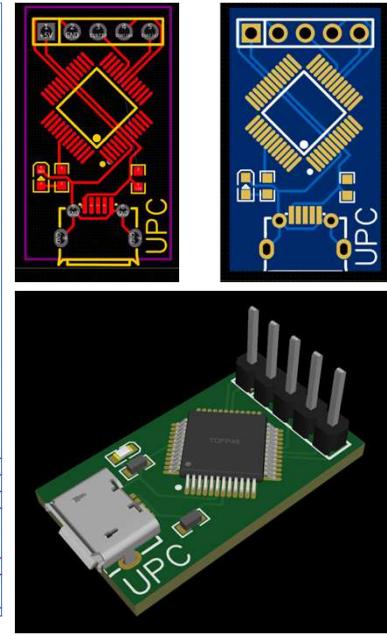
Diseño de PCB empleando EasyEDA

Diseño inicial del curso de Microcontroladores

UPC 2025-1



Schematic	Schematic1		Create at	2025-06-23
Board	Board1		Update at	2025-06-23
Drawn			Page	P1
Reviewed	curiosity_upc			
	Version	Size	Page 1 Total 1	
	V1.0	A4	EasyEDA.com	



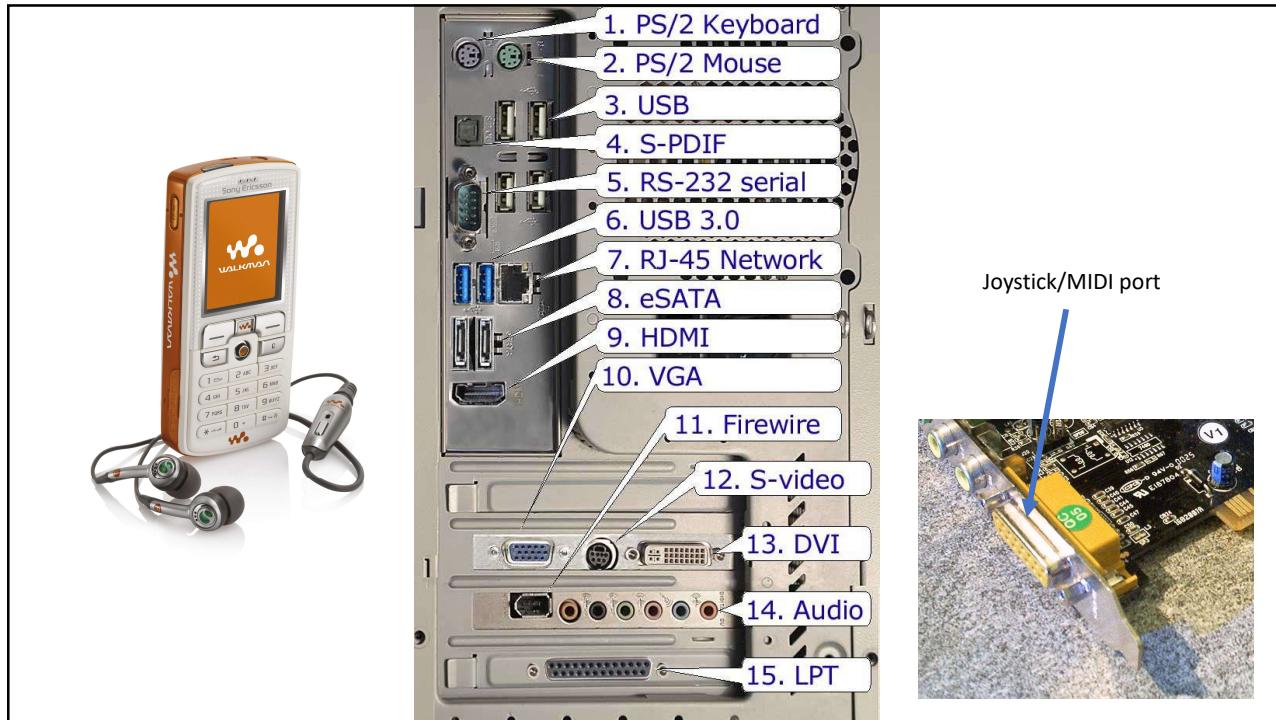
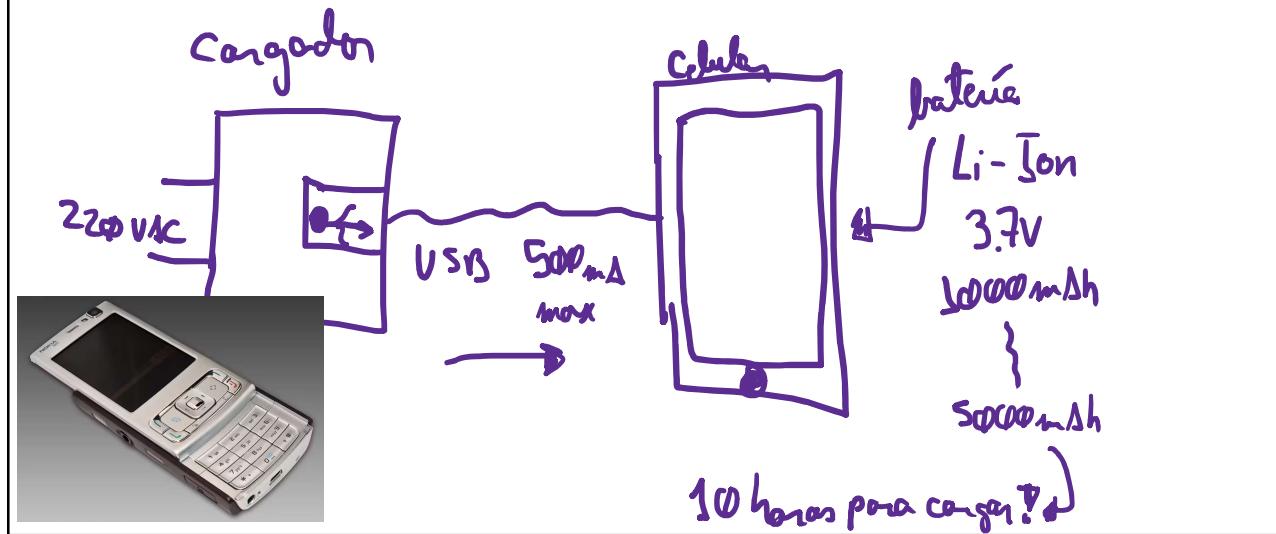
Agenda Semana 12:

Comunicaciones seriales en microcontroladores

- Comunicación UART
 - Niveles lógicos RS232, TTL, RS485
 - Protocolo de comunicación
- El periférico UART del microcontrolador PIC18F57Q43
 - Generador de baudios
 - Modo transmisión
 - Modo recepción

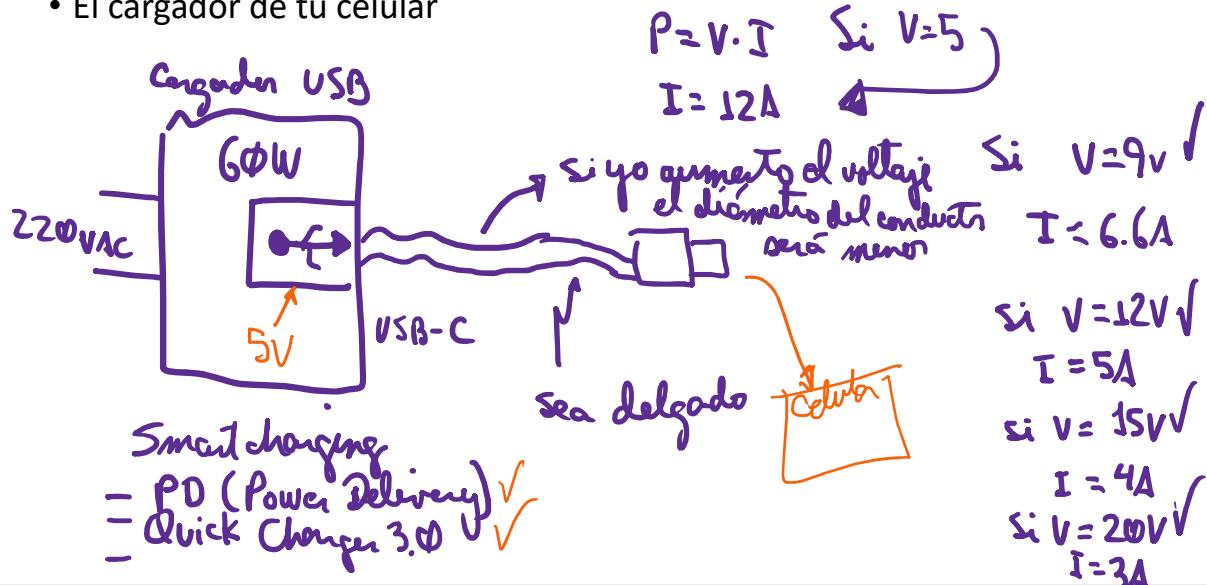
Caso de comunicación serial en la actualidad

- El cargador de tu celular



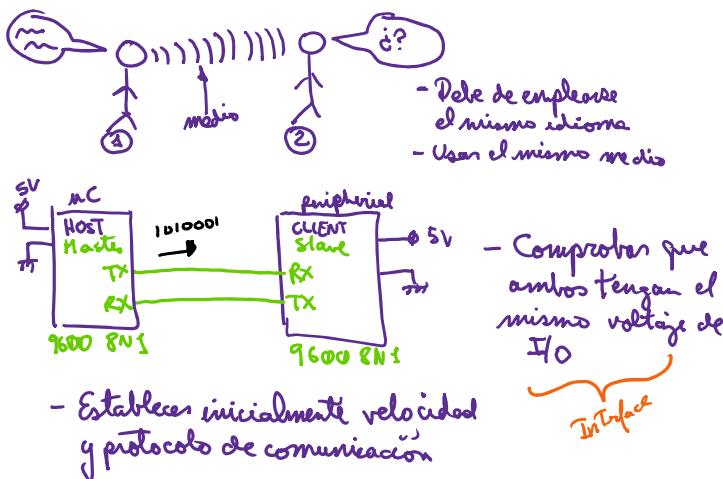
Caso de comunicación serial en la actualidad

- El cargador de tu celular

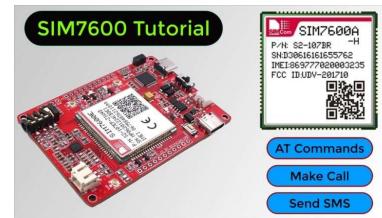
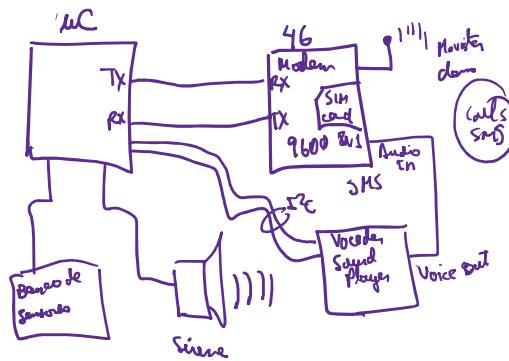


Comunicación asíncrona UART

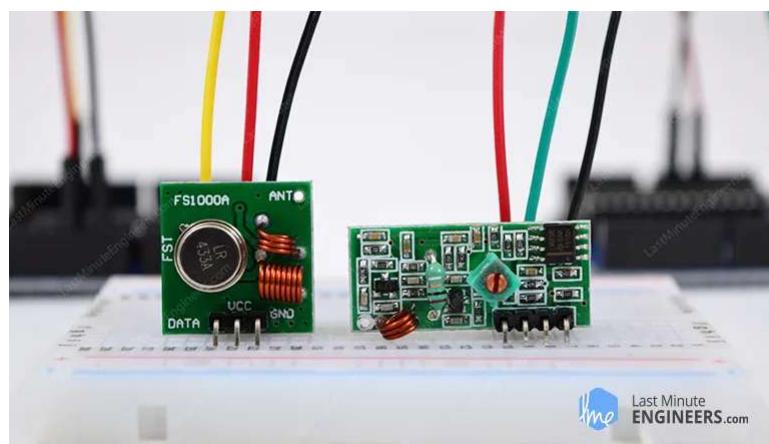
- Asíncrona: No hay una señal dedicada de reloj



Diferentes periféricos externos al microcontrolador



Módulos de radio SAW 434MHz

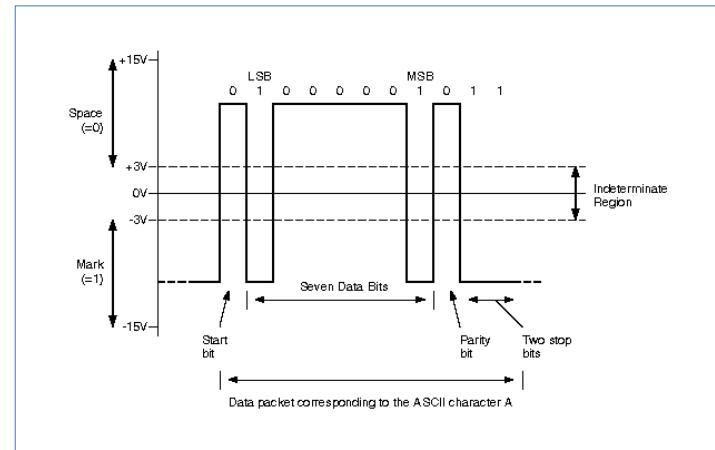
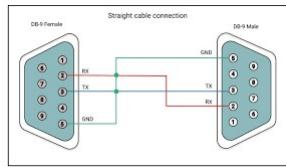
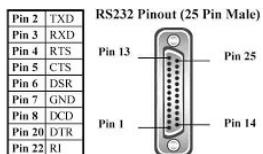


SAW → Surface Acoustic Wave

Sobre niveles de voltaje en comunicación UART

- RS232 (EIA/TIA 232) – Comunicación a distancias medianas (hasta 15 metros)

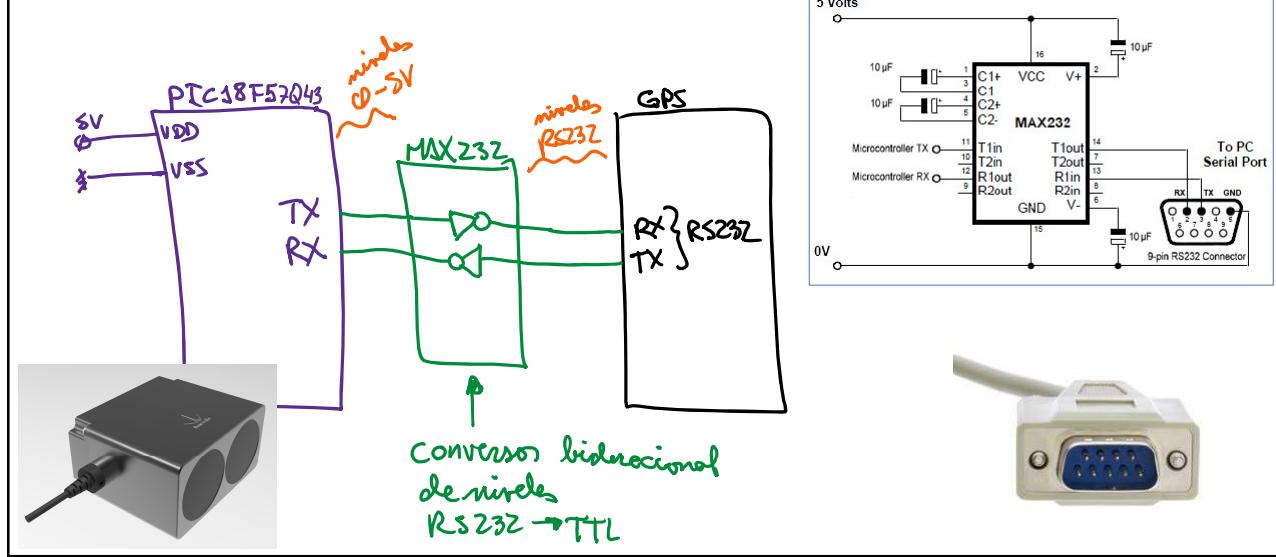
RS232 25 Pin



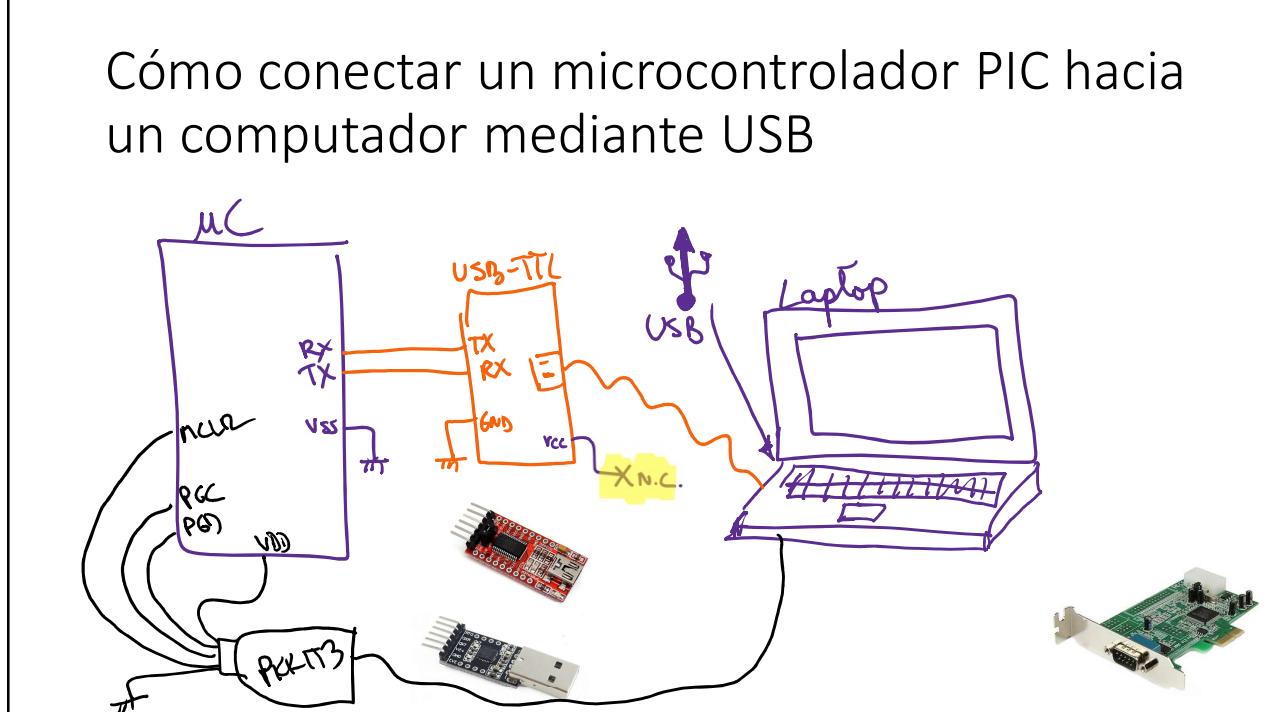
El puerto de consola de equipos de comunicaciones:



¿Cómo hago para conectar un dispositivo RS232 al PIC18F57Q43?



Cómo conectar un microcontrolador PIC hacia un computador mediante USB

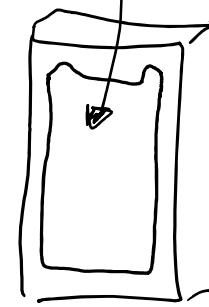
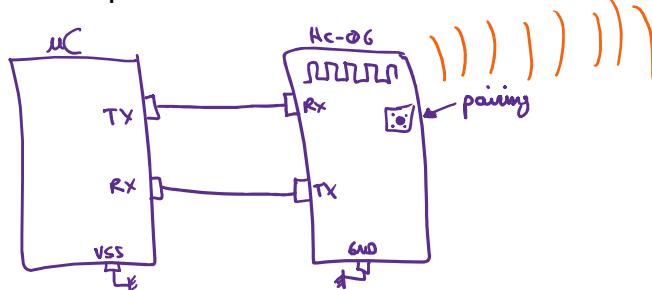


Módulo Bluetooth HC-06



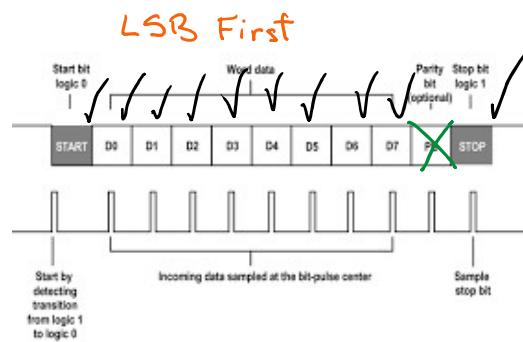
SPP Terminal Serial

- Interface de comunicación UART a niveles TTL (5V)
- Velocidad de comm por defecto: 9600 8N1



Cálculo de la velocidad en comunicación UART

- Formato completo es
 - Velocidad – Protocolo
 - Ej. 9600 - 8N1
- Protocolo común: 8N1
 - 8: El dato es de 8 bits
 - N: no paridad (O:paridad impar, E:paridad par)
 - 1: un bit de stop



- En total se están enviando 10 bits por cada dato de 8 bits

Cálculo de la velocidad en comunicación UART

- Si tengo 9600 8N1:

$$\text{Tiempo de bit: } T_{\text{bit}} = \frac{1}{V_{\text{TX}}} = \frac{1}{9600} = 1.04 \times 10^{-4} \text{ s} \\ = 0.1 \text{ ms}$$

- Si para enviar un dato de 8 bits usamos 10 bits:

$$\text{Tiempo para enviar un dato de 8 bit} = 1.04 \times 10^{-4} (10) = 1.04 \text{ ms.}$$

Ejemplo de cálculo de cuánto tiempo demora en enviar datos vía comm. serial:

- ¿Cuánto demoro en enviar un archivo de 100KByte por una canal de comm. serial a 9600 8N1?

Recordar que 1 byte = 8 bit, 1Kbyte = 1024 bytes

1º Hallar cuantos bits se va a enviar:

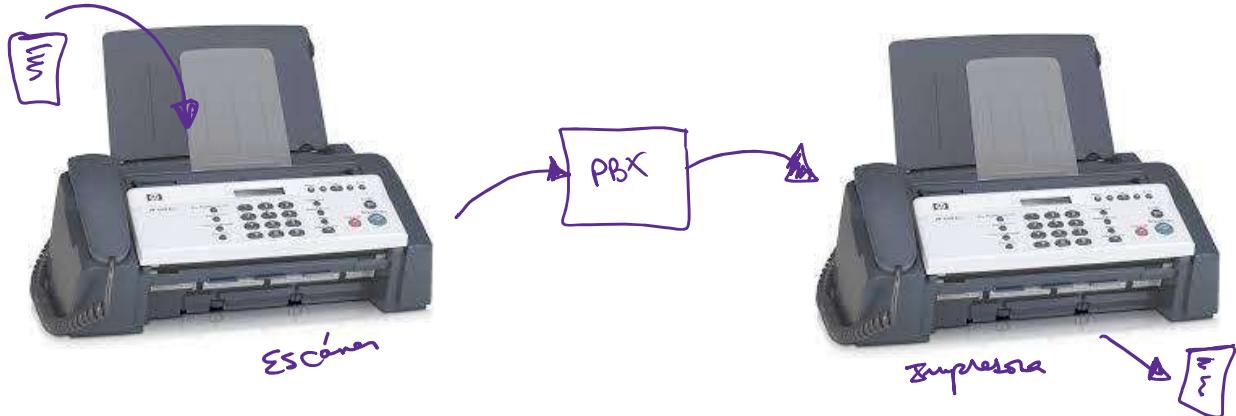
$$100 \times 1024 = 102400 \text{ bytes}$$

Cada byte representará 80 bits enviados $\Rightarrow 1024000$ bits enviados

$$\text{Tiempodebit}_{9600} = 1.04 \times 10^{-4}$$

$$\text{tiempo requerido} = 106.496 \text{ segundos}$$

Recordando las máquinas FAX



- Transmitía el documento ingresado por línea telefónica a 9600 8N1

Ejemplo de cálculo de cuánto tiempo demora en enviar datos vía comm. serial:

- ¿Cuánto demoro en enviar un archivo de música de 4MByte por una canal de comm. serial a 57600 8N1?

5 minutos

Ejemplo de cálculo de cuánto tiempo demora en enviar datos vía comm. serial:

- ¿Cuánto demoro en enviar un archivo de música de 4MByte por una canal de comm. serial a 57600 8N1?

$$\begin{aligned} 4\text{MByte} &\Rightarrow 4\text{K} \times 1024 = 4096 \text{kbytes} \\ &4096\text{K} \times 1024 = 4194304 \text{bytes} \end{aligned}$$

$$\text{Cant. bits a Transmisi\'on: } 4194304 \times 10 = 41943040 \text{ bits}$$

$$\text{Tiempo de bit} \Rightarrow \frac{1}{57600} = 17.361 \mu\text{s}$$

$$\text{Tiempo para transmitir los bits: } 728.177 \text{ s}$$

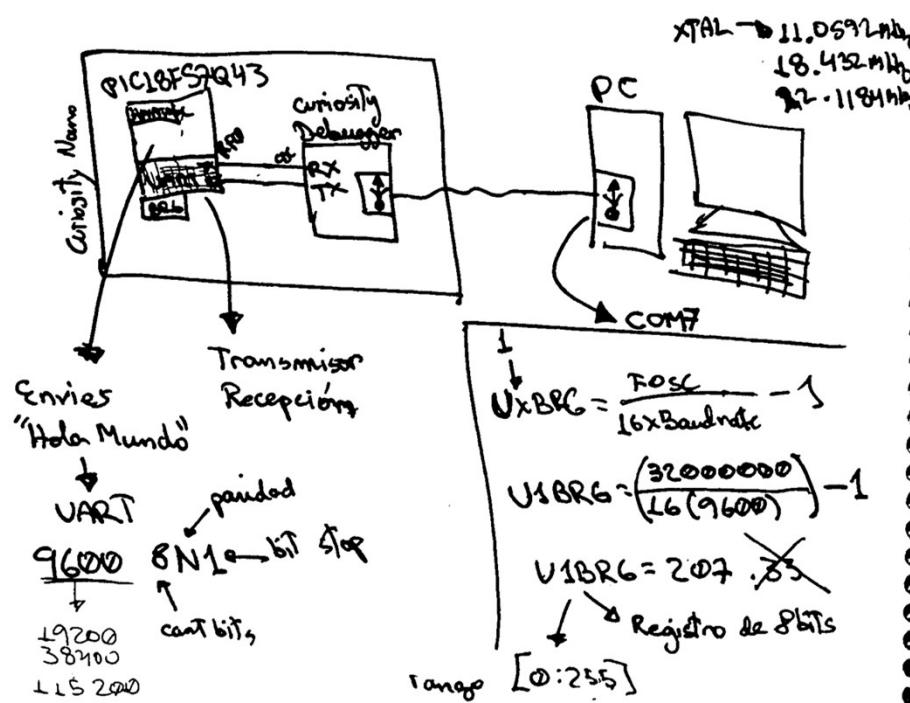
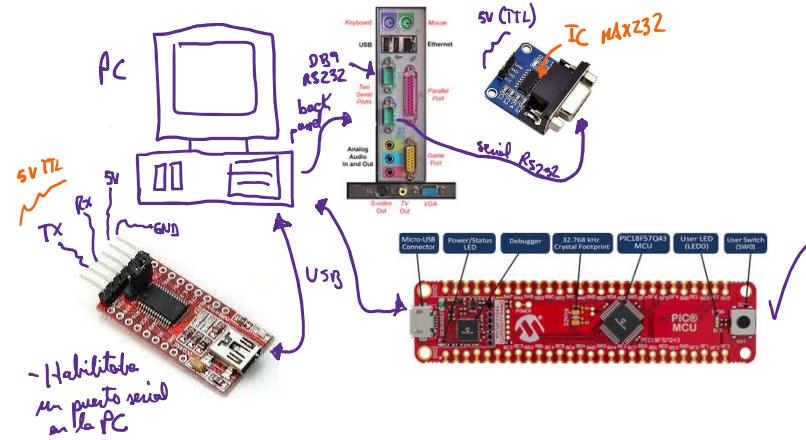
12 minutos y 8.177 segundos

Sobre el UART en el PIC18F57Q43

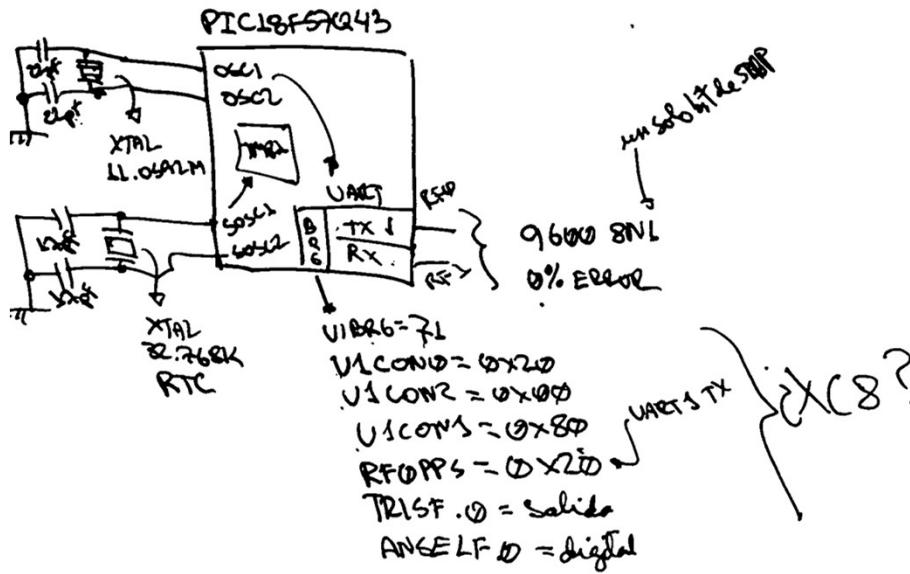
- Revisar capítulo 34 de la hoja técnica del PIC18F57Q43
- Hay 5 módulos UART:
 - 3 full featured (U1, U3, U5)
 - 2 non full featured (U2, U4)

Comunicación serial UART con el PIC18F57Q43

- Curiosity Nano -> PC



Solución con 0% Error en UART



Ejemplo: Comunicación serial UART

- Circuito de prueba

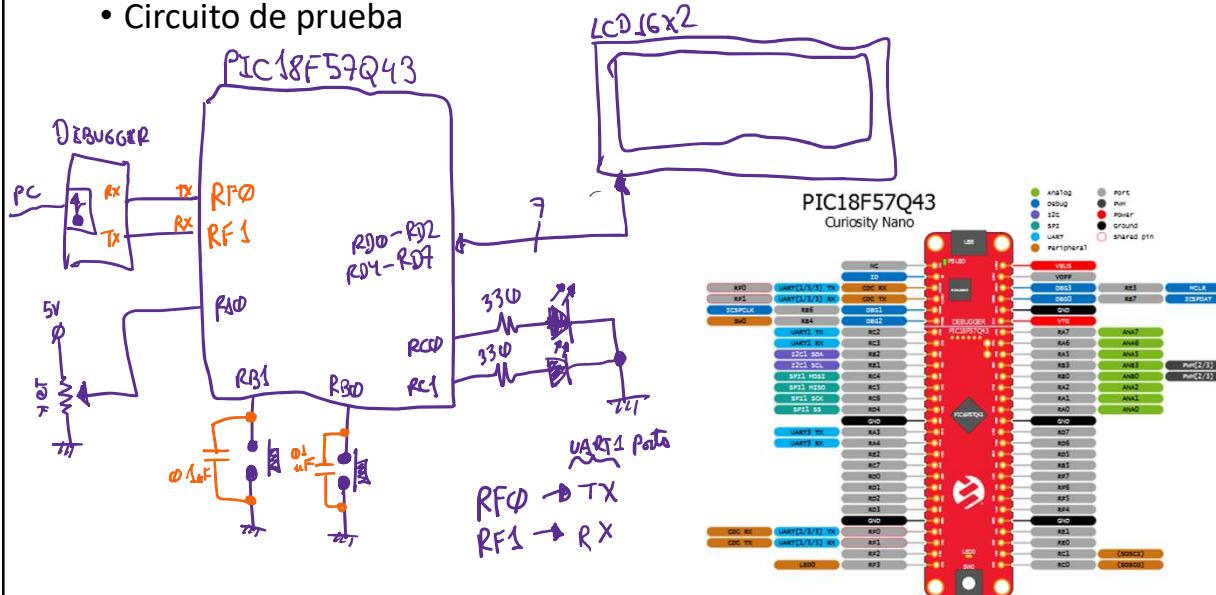
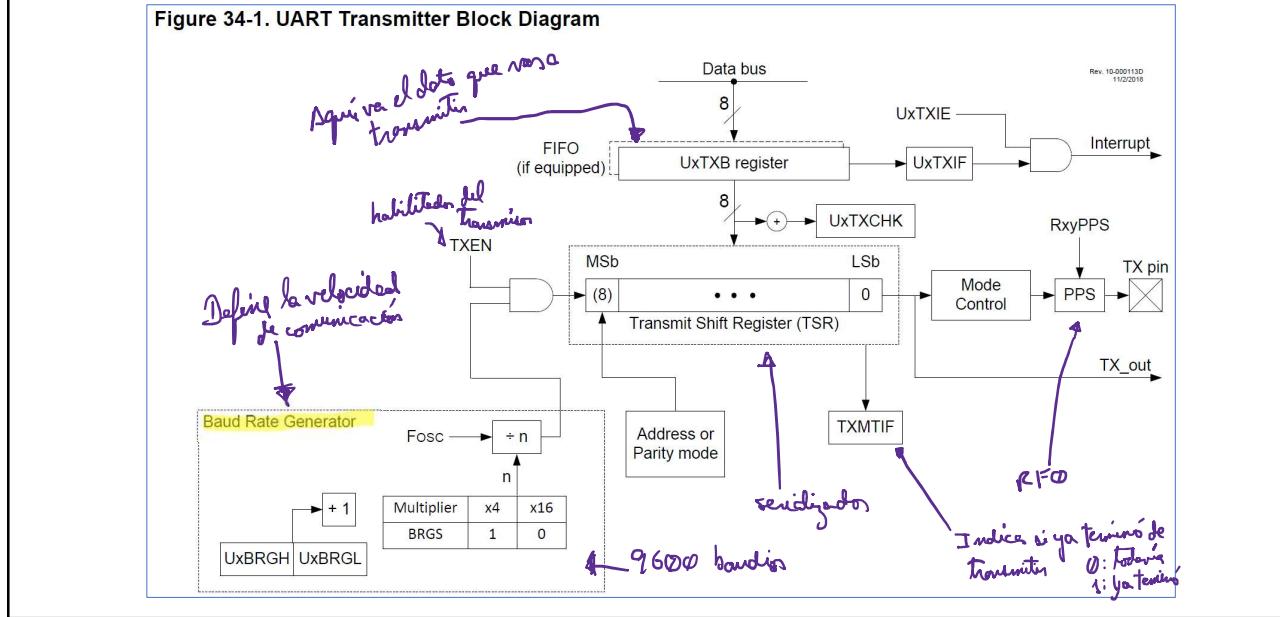


Diagrama de bloques del transmisor del UART

Figure 34-1. UART Transmitter Block Diagram



Fórmula para calcular el BRG:

The UART Baud Rate equals $[Fosc * (1 + (BRGS * 3))] / [(16 * (BRG + 1))]$

Tenemos que despejamos BRG

$$\text{Baud rate} = 9600 \quad Fosc = 32\text{MHz}$$

$$BRGS = 4$$

$$9600 = \frac{32 \times 10^6 (1 + 4 \times 3)}{16 \times (BRG + 1)}$$

$$BRG + 1 = \frac{[32 \times 10^6 (1)]}{16 \times 9600} \quad BRG = \frac{(32000000)}{153600} - 1 = 207.3333$$

Para 9600:

$$BRG + 1 = \frac{32 \times 10^6 (1)}{16 \times 9600} \quad BRG = \frac{32000000}{16 \times 9600} - 1$$

$$BRG = \frac{32000000}{307200} - 1 \quad BRG = 103$$

Para 38400:

$$BRG = \frac{32000000}{16 \times 38400} - 1$$

$$BRG = \frac{32000000}{614400} - 1$$

$$BRG = 51$$

Registro UxCON0

- Se configura el BRGS, los habilitadores de TX y RX y el modo de trabajo del UART1 (normalmente acíncrono y 8 bits de datos)

Name: UxCON0	
Address: 0x2AB,0x2BE,0x2D1,0x2E4,0x2F7	
UART Control Register 0	
Bit	
Access	BRGS R/W
Reset	ABDEN R/W
	TXEN R/W
	RXEN R/W
	MODE[3:0] R/W
	SENDB R/W
	TRWD R/W
	= 0x20
Bit 7 – BRGS Baud Rate Generator Speed Select	
Value	Description
1	Baud Rate Generator is high speed with 4 baud clocks per bit
0	Baud Rate Generator is normal speed with 16 baud clocks per bit
Bit 6 – ABDEN Auto-Baud Detect Enable⁽¹⁾	
Value	Description
1	Auto-baud is enabled. Receiver is waiting for Sync character (0x55).
0	Auto-baud is not enabled or auto-baud is complete
Bit 5 – TXEN Transmit Enable Control⁽²⁾	
Value	Description
1	Transmit is enabled. TX output pin drive is forced on when transmission is active, and controlled by PORT TRIS control when transmission is idle.
0	Transmit is disabled. TX output pin drive is controlled by PORT TRIS control.
Bit 4 – RXEN Receive Enable Control⁽³⁾	
Value	Description
1	Receiver is enabled
0	Receiver is disabled
Bits 3:0 – MODE[3:0] UART Mode Select⁽¹⁾	
Value	Description
1111	Reserved
1101	
1100	LIN Host/Client mode ⁽⁴⁾
1011	LIN Client Only mode ⁽⁴⁾
1010	DMX mode ⁽⁴⁾
1001	DALI Control Gear mode ⁽⁴⁾
1000	DALI Control Device mode ⁽⁴⁾
0111	Reserved
0101	
0100	Asynchronous 9-bit UART Address mode. 9th bit: 1 = address, 0 = data
0011	Asynchronous 8-bit UART mode with 9th bit even parity
0010	Asynchronous 8-bit UART mode with 9th bit odd parity
0001	Asynchronous 7-bit UART mode
0000	Asynchronous 8-bit UART mode

Registro UxCON1

- Se configura el habilitador del UARTx

Name: UxCON1	
Address: 0x2AC,0x2BF,0x2D2,0x2E5,0x2F8	
UART Control Register 1	
Bit	
Access	ON R/W
Reset	0
	WUE R/W/HC
	RXBIMD R/W
	BRKOVVR R/W
	SENDB R/W/HC
	= 0x80
Bit 7 – ON Serial Port Enable	
Value	Description
1	Serial port enabled
0	Serial port disabled (held in Reset)
Bit 4 – WUE Wake-Up Enable	
Value	Description
1	Receiver is waiting for falling RX input edge which will set the UxIF bit. Cleared by hardware on wake-up event. Also requires the UxIE bit of PIEL to enable wake.
0	Receiver operates normally
Bit 3 – RXBIMD Receive Break Interrupt Mode Select	
Value	Description
1	Set RXBKIF immediately when RX in has been low for the minimum Break time
0	Set RXBKIF on rising RX input after RX in has been low for the minimum Break time
Bit 1 – BRKOVVR Send Break Software Override	
Value	Description
1	TX output is forced to non-idle state
0	TX output is driven by transmit shift register
Bit 0 – SENDB Send Break Control⁽¹⁾	
Value	Description
1	Output Break upon UxTXB write. Written byte follows Break. Bit is cleared by hardware.
0	Break transmission completed or disabled

Registro U1CON2

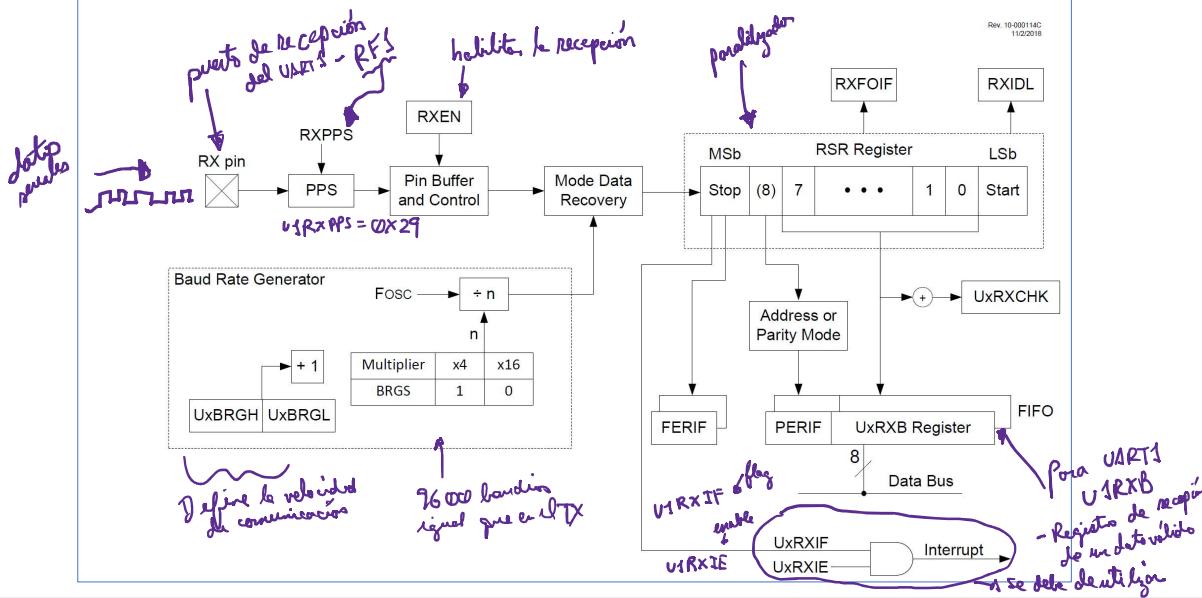
- Lo vamos a dejar con los valores por defecto (0x00)

Configuración del UART1 para transmitir un dato:

- Primero: Puertos de E/S:
 - En el Curiosity Nano RF0 es el TX, RF1 es el RX (configurar TRIS y ANSEL)
- Segundo: Establecer velocidad con U1BRG
- Tercero: Configurar registros U1CONx:
 - U1CON0: BRGS (normal/high), TX enable, RX enable, UART mode
 - U1CON1: ON (serial port enable)
 - U1CON2: déjalo con los valores por defecto
- Cuarto: El PPS para TX
 - En el Curiosity Nano: RF0PPS = 0x20;
- Quinto: Para transmitir un dato de 8 bits
 - Colocar el dato en U1TXB
 - Esperar a que se termine de transmitir (U1ERRIR bit TXMTIF = 1)

Diagrama de bloques del receptor del UART

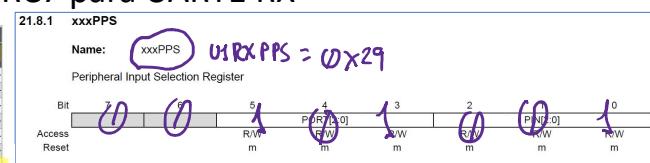
Figure 34-2. UART Receiver Block Diagram



Configuración del PPS de entrada del UART1 RX con el puerto RF1

- Esta por defecto el RC7 para UART1 RX

Peripheral	PPS Input Register	Definitive Pin Selection at POR	Register Reset Value at POR	28-Pin Devices	40-Pin Devices	Available Input Port
SP11 Client Select	SP11SSPPS	RA5	'b000 101	A — C A — D A	B — C B D B D B	
SP21 Clock	SP12SCKPPS	RB3	'b001 011	— B — C — D —	— B — C B D B D B	
SP12 Data	SP12SDPPS	RB2	'b001 010	— B — C — D —	— B — C B D B D B	
SP12 Client Select	SP12SSPPS	RA4	'b000 100	A — C A — D A	B — C B C B C B C	
I2C1 Clock	I2C1SCLPPS ⁽¹⁾	RC3	'b010 011	— B — C — D —	— B — C B C B C B C	
I2C1 Data	I2C1SDAPPSS ⁽¹⁾	HC4	'b010 100	— B — C — D —	— B — C B C B C B C	
UART1 Receive	U1RXPPS	RC7	'b010 111	— B — C — D —	— B — C B C B C B C	
UART1 Clear to Send	U1CTSPS	RC6	'b010 110	— B — C — D —	— B — C B C B C B C	
UART2 Receive	U2RXPPS	RB7	'b001 111	— B — C — D —	— B — C B D B D B	
UART2 Clear to Send	U2CTSPS	RB6	'b001 110	— B — C — D —	— B — C B D B D B	
UART3 Receive	USRXPPS	RA7	'b000 111	A B — A B — A	B C — B D — B	
UART3 Clear to Send	USCTSPS	RA6	'b000 110	A B — A B — A	B C — B D — B	
UART4 Receive	U4RXPPS	RB5	'b001 101	— B — C — D —	— B — C B D B D B	
UART4 Clear to Send	U4CTSPS	RB4	'b001 100	— B — C — D —	— B — C B D B D B	
UART5 Receive	USRXPPS	RA5	'b000 101	A — C A — C —	A — C A — C —	
UART5 Clear to Send	USCTSPS	RA4	'b000 100	A — C A — C —	A — C A — C —	



Bits 5:3 – PORT[2:0] Peripheral Input PORT Selection⁽¹⁾
See the [PPS Input Selection Table](#) for the list of available Ports and default pin locations

PORT		Selection
101		PORTF
100		PORTE
011		PORTD
010		PORTC
001		PORTB
000		PORTA

Reset States: POR = mmm
All other Resets = uuu

Bits 2:0 – PIN[2:0] Peripheral Input PORT Pin Selection ⁽²⁾	
Reset States: POR = 000	
All other Resets = 000	
Value	Description
111	Peripheral input is from PORTTx Pin 7 (Rx7)
110	Peripheral input is from PORTTx Pin 6 (Rx6)
101	Peripheral input is from PORTTx Pin 5 (Rx5)
100	Peripheral input is from PORTTx Pin 4 (Rx4)
011	Peripheral input is from PORTTx Pin 3 (Rx3)
010	Peripheral input is from PORTTx Pin 2 (Rx2)
001	Peripheral input is from PORTTx Pin 1 (Rx1)
000	Peripheral input is from PORTTx Pin 0 (Rx0)

Registro U1CON0

Name: UxCON0 Address: 0x2AB,0x2BE,0x2D1,0x2E4,0x2F7							
UART Control Register 0							
Bit	BRGS	ABDEN	TXEN	RXEN	MODE[3:0]		
Access	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0
Bit 7 – BRGS Baud Rate Generator Speed Select							
Value	Description						
1	Baud Rate Generator is high speed with 4 baud clocks per bit						
0	Baud Rate Generator is normal speed with 16 baud clocks per bit						
Bit 6 – ABDEN Auto-Baud Detect Enable⁽³⁾							
Value	Description						
1	Auto-baud is enabled. Receiver is waiting for Sync character (0x55).						
0	Auto-baud is not enabled or auto-baud is complete						
Bit 5 – TXEN Transmit Enable Control⁽²⁾							
Value	Description						
1	Transmit is enabled. TX output pin drive is forced on when transmission is active, and controlled by PORT TRIS control when transmission is Idle.						
0	Transmit is disabled. TX output pin drive is controlled by PORT TRIS control.						
Bit 4 – RXEN Receive Enable Control⁽²⁾							
Value	Description						
1	Receiver is enabled						
0	Receiver is disabled						
Bits 3:0 – MODE[3:0] UART Mode Select⁽¹⁾							
Value	Description						
1111	Reserved						
1101							
1100	LIN Host/Client mode ⁽⁴⁾						
1011	LIN Client Only mode ⁽⁴⁾						
1010	DMX mode ⁽⁴⁾						
1001	DALI Control Gear mode ⁽⁴⁾						
1000	DALI Control Device mode ⁽⁴⁾						
0111	Reserved						
0101							
0100	Asynchronous 9-bit UART Address mode. 9th bit: 1 = address, 0 = data						
0011	Asynchronous 8-bit UART mode with 9th bit even parity						
0010	Asynchronous 8-bit UART mode with 9th bit odd parity						
0001	Asynchronous 7-bit UART mode						
0000	Asynchronous 8-bit UART mode						

Ubicación de la interrupción de recepción del UART1:

UX04A1	PIE3	I.0	TMR0IE	COP0IE	TMR1GIE	TMR1IE	TMR2IE	SPI1IE	SPI1TXIE	SPI1RXIE
0x04A2	PIE4	7:0	PWM1IE	PWM1PIE		U1IE	U1IE	U1TXIE	U1RXIE	
0x04A3	PIE5	7:0	PWM2IE	PWM2PIE	TMR3GIE	TMR3IE		SPI2IE	SPI2TXIE	SPI2RXIE
0x04A4	PIE6	7:0	DMA2AIE	DMA2ORIE	DMA2DCNTIE	DMA2SCNTIE	NCO0IE	CWG1IE	CLC2IE	INT1IE
0x04A5	PIE7	7:0	PWM3IE	PWM3PIE	CLC3IE		I2C1IE	I2C1IE	I2C1TXIE	I2C1RXIE
0x04A6	PIE8	7:0	SCANIE	CCP2IE	TMR5GIE	TMR5IE	U2IE	U2IE	U2TXIE	U2RXIE
0x04A7	PIE9	7:0		CLC4IE			U3IE	U3IE	U3TXIE	U3RXIE
0x04A8	PIE10	7:0	DMA3AIE	DMA3ORIE	DMA3DCNTIE	DMA3SCNTIE	NCO2IE	CWG2IE	CLC5IE	INT2IE
0x04A9	PIE11	7:0	DMA4AIE	DMA4ORIE	DMA4DCNTIE	DMA4SCNTIE	TMR4IE	CWG3IE	CLC6IE	CCP3IE
0x04AA	PIE12	7:0	DMA5AIE	DMA5ORIE	DMA5DCNTIE	DMA5SCNTIE	U4IE	U4IE	U4TXIE	U4RXIE
0x04AB	PIE13	7:0	DMA6AIE	DMA6ORIE	DMA6DCNTIE	DMA6SCNTIE	U5IE	U5IE	U5TXIE	U5RXIE
0x04AC	PIE14	7:0					NCO3IE	CM2IE	CLC7IE	
0x04AD	PIE15	7:0					TMR6IE	CRCIE	CLC8IE	NVME
0x04AE	PIR0	7:0	IOCF		CLC1IF		CSWIF	OSFIF	HLVDIF	SWIF
0x04AF	PIR1	7:0	SMT1PWAIF	SMT1PRAIF	SMT1IF	CM1IF	ACTIF	ADIF	ZCDIF	INTOIF
0x04B0	PIR2	7:0	DMA1AIF	DMA1ORIF	DMA1DCNTIF	DMA1SCNTIF				ADTF
0x04B1	PIR3	7:0	TMR0IF	CCP1IF	TMR1GIF	TMR1IF	TMR2IF	SPI1IF	SPI1TXIF	SPI1RXIF
0x04B2	PIR4	7:0	PWM1IF	PWM1PIF			U1IF	U1EIF	U1TXIF	U1RXIF
0x04B3	PIR5	7:0	DWMA0IF	DWMA0PIF	TMD2GIF	TMD2IF		SPID1IF	SPID1TXIF	SPID1RXIF

Configuración adicional del UART1 para receptionar un dato:

- Primero: Estas configuraciones son adicionales a lo que se establecieron en la transmisión
- Segundo: Cerciorarse que el RX esté enabled en el U1CON0
- Tercero: Habilitar la interrupción de la recepción del UART1:
 - Habilitador U1RXIE se encuentra en PIE4 y bandera U1RXIF se encuentra en PIR4
 - No olvidar habilitar el GIE que esta en INTCON0.
- Cuarto: El PPS para RX
 - En el Curiosity Nano: U1RXPPS = 0x29
- Quinto: Para recibir un dato de 8 bits
 - Definir la función de interrupción de la recepción del UART1

```
void __interrupt(irq(IRQ_U1RX)) U1RX_ISR(void){
    PIR4bits.U1RXIF = 0;           //bajamos bandera de RX de UART1
```

Agenda Semana 13:

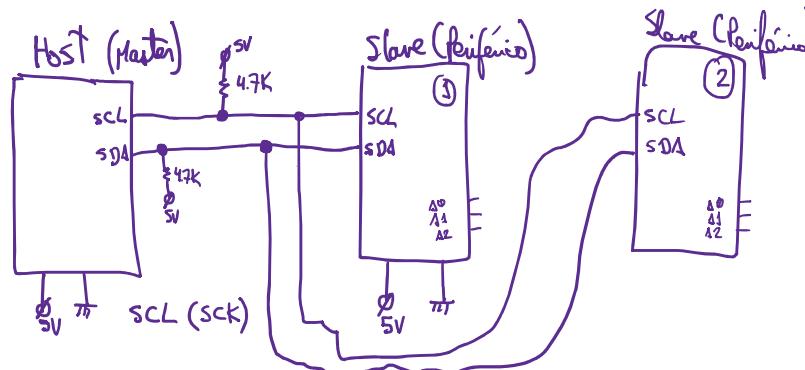
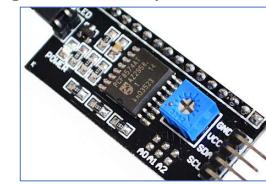
Comunicaciones seriales en microcontroladores

- Comunicación I2C
 - Interface eléctrica
 - Protocolo de comunicación
 - Periféricos externos con capacidad I2C
- El I2C del PIC18F57Q43
 - Configuración

Si tienen alguna consulta, levantan la mano para poder atenderlos.

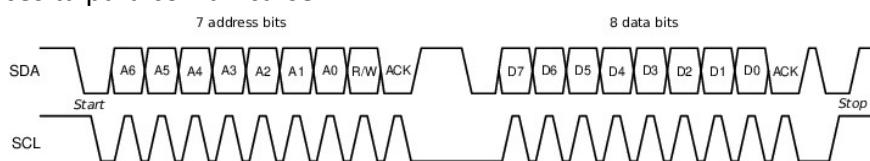
Comunicación I2C (initialmente fue IIC)

- Comunicación desarrollada para la comunicación entre circuitos integrados. Philips Semiconductor (NXP).
- Comunicación serial síncrona (presencia de señal de reloj)
- I2C – I2S -> Sonido
- I3C – Evolución de las comunicaciones I2C
- Podemos construir una red de dispositivos (topología bus)



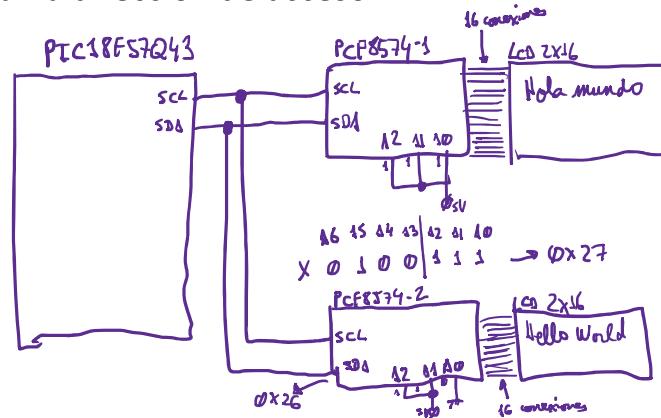
Comunicación I2C

- Direcciones de 7 bits ó 10 bits
- Frecuencia de reloj de 100Khz (legacy o normal), 400Khz (fast mode), 3.4KHz (high-speed mode), 5MHz (ultra-fast mode).
- Puede funcionar en modo HOST (master) o en modo CLIENT (slave)
- En el I2C tenemos condiciones (comandos u órdenes):
 - Condición **START**: Inicio de la comunicación I2C.
 - Condición **RESTART**: Resetear la línea de comunicación, generalmente se emplea para cambiar entre escritura y lectura dentro de un evento de comunicación de un periférico.
 - Condición **STOP**: Término de la comunicación I2C.
 - **ACK** (Acknowledge): Evento de confirmación.
 - **NOACK** (No Acknowledge): Evento de no confirmación.
- Se debe de revisar la hoja técnica del periférico I2C para ver la trama I2C que necesita para comunicarse



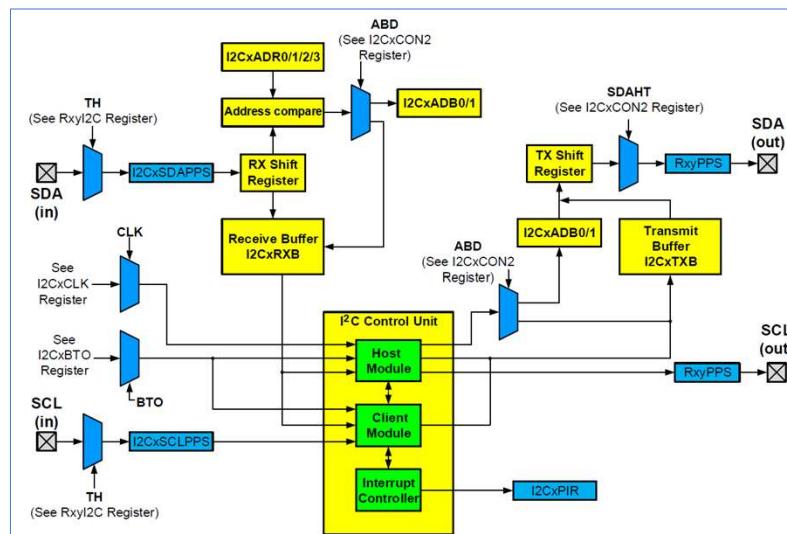
Consideraciones en I2C

- En modo de direcciones de siete bits, en teoría se puede conectar hasta 128 dispositivos, pero en la práctica dependerá del fabricante del periférico.
- En el caso del PCF8574, tenemos disponibles tres bits para poder personalizar la dirección de acceso



El módulo I2C del PIC18F57Q43

- Revisar capítulo 36 de la hoja técnica:



El módulo I2C del PIC18F57Q43

- Para obtener 100KHz de frecuencia de trabajo (modo normal):
 - Se emplea MFINTOSC ya que al ser de 500KHz y configurando FME (Fast Mode Enabled) a cero se obtienen los 100KHz:

Equation 36-1. SCL Frequency (FME = 0)

$$f_{SCL} = \frac{f_{I2CxCLK}}{5}$$

Example:

- I2CxCLK: MFINTOSC (500 kHz)
- FME: FME = 0

El módulo I2C del PIC18F57Q43

- Escritura de datos

Figure 36-26. 7-Bit Host Write Diagram

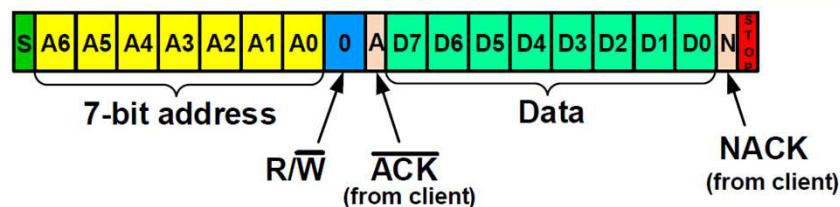
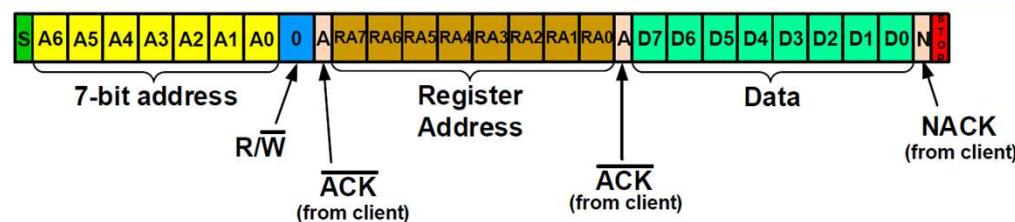


Figure 36-27. 7-Bit Host Write Diagram (To a Specific Memory/Register Location)



El módulo I2C del PIC18F57Q43

- Procedimiento para hacer una transmisión:

36.4.2.7.1 Host Transmission (7-Bit Addressing Mode)

The following section describes the sequence of events that occur when the module is transmitting data in 7-bit Addressing mode:

- Depending on the configuration of the Address Buffer Disable (ABD) bit, one of two methods may be used to begin communication:
 - When ABD is clear (ABD = 0), the address buffer, **I2CxADB1**, is enabled. In this case, the 7-bit client address and R/W bit are loaded into **I2CxADB1**, with the R/W bit clear (R/W = 0). The number of data bytes are loaded into **I2CxCNT** and the first data byte is loaded into **I2CXTXB**. After these registers are loaded, software must set the Start (S) bit to begin communication. Once the S bit is set, host hardware waits for the Bus Free (BFRE) to be set before transmitting the Start condition to avoid bus collisions.
 - When ABD is set (ABD = 1), the address buffer is disabled. In this case, the number of data bytes are loaded into **I2CxCNT**, and the client's 7-bit address and R/W bit are loaded into **I2CXTXB**. A write to **I2CXTXB** will cause host hardware to automatically issue a Start condition once the bus is idle (BFRE = 1). Software writes to the Start bit are ignored.
- Host hardware waits for BFRE to be set, then shifts out the Start condition. Module hardware sets the Host Mode Active (HMA) bit and the Start Condition Interrupt Flag (SCIF). If the Start Condition Interrupt Enable (SCIE) bit is set, the generic I2CxIF is also set.
- Host hardware transmits the 7-bit client address and R/W bit.
- If upon the 8th falling edge of SCL, **I2CXTXB** is empty (Transmit Buffer Empty Status (TXBE) = 1), **I2CxCNT** is non-zero (I2CxCNT ≠ 0), and the Clock Stretching Disable (CSD) bit is clear (CSD = 0):
 - The PC Transmit Interrupt Flag (I2CXTXIF) is set. If the PC Transmit Interrupt Enable (I2CXTXIE) bit is also set.
 - The Host Data Request (MDR) bit is set, and the clock is stretched, allowing time for software to load **I2CXTXB** with new data. Once **I2CXTXB** has been written, hardware releases SCL and clears MDR.
- Hardware transmits the 9th clock pulse and waits for an ACK/NACK response from the client. If the host receives an ACK, module hardware transfers the data from **I2CXTXB** into the transmit shift register, and **I2CxCNT** is decremented by one. If the host receives a NACK, hardware will attempt to issue a Stop condition. If the clock is currently being stretched by a client, the host must wait until the bus is free before issuing the Stop.
- Host hardware checks **I2CxCNT** for a zero value. If **I2CxCNT** is zero:
 - If ABD is clear (ABD = 0), host hardware issues a Stop condition, or sets MDR if the Restart Enable (RSEN) bit is set and waits for software to set the Start bit to issue a Restart condition. CNTIF is set.
 - If ABD is set (ABD = 1), host hardware issues a Stop condition, or sets MDR if RSEN is set and waits for software to load **I2CXTXB** with a new client address. CNTIF is set.
- Host hardware transmits the data byte.
- If upon the 8th falling edge of SCL **I2CXTXB** is empty (TXBE = 1), **I2CxCNT** is non-zero (I2CxCNT ≠ 0), and CSD is clear (CSD = 0):
 - I2CXTXIF is set. If the I2CXTXIE bit is also set, the generic I2CxIF is also set.
 - The MDR bit is set and the clock is stretched, allowing time for software to load **I2CXTXB** with new data. Once **I2CXTXB** has been written, hardware releases SCL and clears MDR.
- If TXBE is set (TXBE = 1) and **I2CxCNT** is zero (I2CxCNT = 0):
 - I2CxIF is NOT set.
 - CNTIF is set.
 - Host hardware issues a Stop condition, setting PCIF.
- Repeat Steps 5 – 9 until all data has been transmitted.

El módulo I2C del PIC18F57Q43

- Lectura de datos:

Figure 36-23. 7-Bit Host Read Diagram

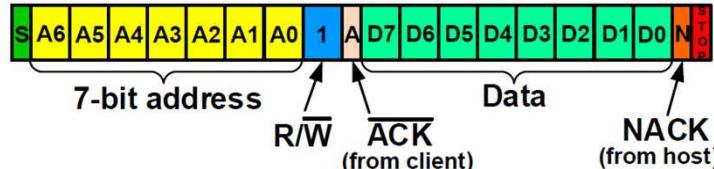
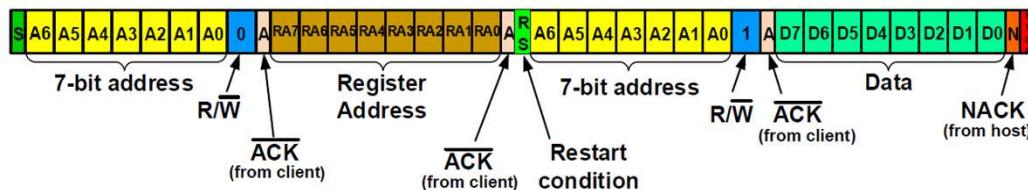


Figure 36-24. 7-Bit Host Read Diagram (From a Specific Memory/Register Location)



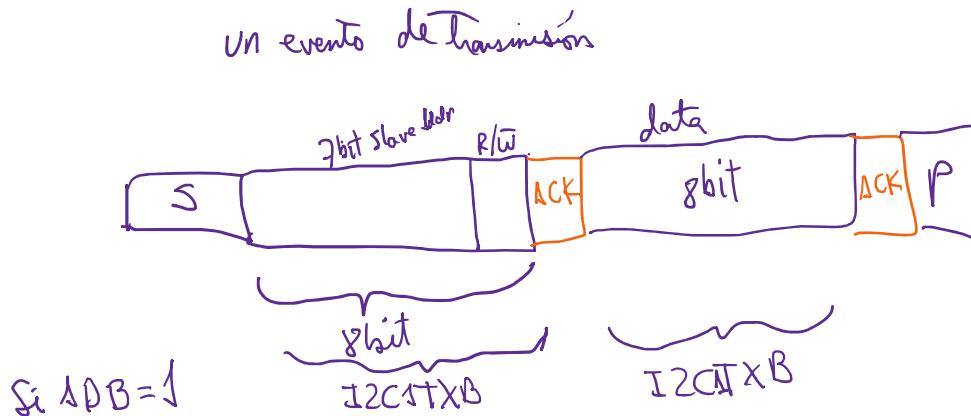
El módulo I2C del PIC18F57Q43

- Procedimiento para hacer una recepción:

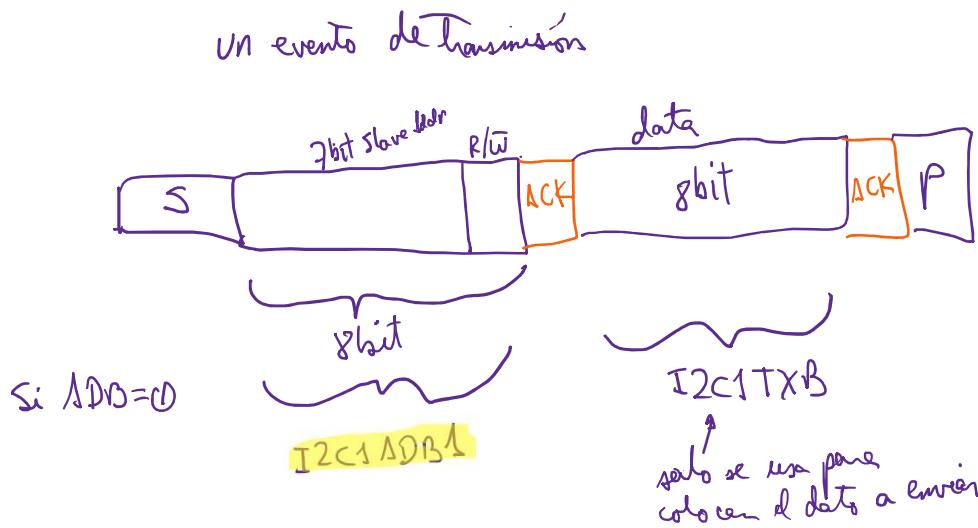
36.4.2.7.2 Host Reception (7-Bit Addressing Mode)
The following section describes the sequence of events that occur when the module is receiving data in 7-bit Addressing mode:

1. Depending on the configuration of the Address Buffer Disable (ADB) bit, one of two methods may be used to begin communication:
 - a. When ADB is clear (ADB = 0), the address buffer, I2CxADB1, is enabled. In this case, the 7-bit client address and R/W bit are loaded into I2CxADB1, with the R/W bit set (R/W = 1). The number of expected received data bytes are loaded into I2CxCNT. After these registers are loaded, software must set the Start (S) bit to begin communication. Once the S bit is set, host hardware waits for the Bus Free (BFRE) bit to be set before transmitting the Start condition to avoid bus collisions.
 - b. When ADB is set (ADB = 1), the address buffer is disabled. In this case, the number of expected received data bytes are loaded into I2CxCNT, and the client's 7-bit address and R/W bit are loaded into I2CxTXB. A write to I2CxTXB will cause host hardware to automatically issue a Start condition once the bus is idle (BFRE = 1). Software writes to the Start bit are ignored.
2. Host hardware waits for BFRE to be set, then shifts out the Start condition. Module hardware sets the Host Mode Active (MMA) bit and the Start Condition Interrupt Flag (SCIF). If the Start Condition Interrupt Enable (SCIE) bit is set, the generic I2CxIF is also set.
3. Host hardware transmits the 7-bit client address and R/W bit.
4. Host hardware samples SCL to determine if the client is stretching the clock, and continues to sample SCL until the line is sampled high.
5. Host hardware transmits the 9th clock pulse, and receives the ACK/NACK response from the client. If an ACK is received, host hardware receives the first seven bits of the data byte into the receive shift register. If a NACK is received, hardware sets the NACK Detect Interrupt Flag (NACKIF), and:
 - a. **ADB = 0:** Host generates a Stop condition, or sets the MDR bit (if RSEN is also set) and waits for software to set the Start bit to generate a Restart condition.
 - b. **ADB = 1:** Host generates a Stop condition, or sets the MDR bit (if RSEN is also set) and waits for software to load a new address into I2CxTXB. Software writes to the Start bit are ignored.
- If the NACK Detect Interrupt Enable (NACKIE) is also set, hardware sets the generic I2CxEIF bit.
6. If previous data remains in the I2C Receive Buffer (I2CxRB) when the first seven bits of the new byte are received into the receive shift register (RXBF = 1), the MDR bit is set (MDR = 1), and the clock is stretched after the 7th falling edge of SCL. This allows the host time to read I2CxRB, which clears the RXBF bit, and prevents receive buffer overflows. Once RXBF is clear, hardware releases SCL.
7. The host clocks in the 8th bit of the data byte into the receive shift register, then transfers the full byte into I2CxRB. Host hardware sets the I2C Receive Interrupt Flag (I2CxRIF) and RXBF, and if the I2C Receive Interrupt Enable (I2CxRIE) is set, the generic I2CxIF is also set. Finally, I2CxCNT is decremented by one.
8. Host hardware checks I2CxCNT for a zero value. If I2CxCNT is non-zero (I2CxCNT ≠ 0), hardware transmits the value of the Acknowledge Data (ACKDT) bit as the acknowledgement response to the client. It is up to user software to properly configure ACKDT. In most cases, ACKDT should be clear (ACKDT = 0), which indicates an ACK response.
- If I2CxCNT is zero (I2CxCNT = 0), hardware transmits the value of the Acknowledge End of Count (ACKCNT) bit as the acknowledgement response to the client. CNTIF is set, and host hardware either issues a Stop condition or a Restart condition. It is up to user software to properly configure ACKCNT. In most cases, ACKCNT should be set (ACKCNT = 1), which indicates a NACK response. When hardware detects a NACK on the bus, it automatically issues a Stop condition. If a NACK is not detected, the Stop will not be generated, which may lead to a stalled bus condition.
9. Host hardware receives the first seven bits of the next data byte into the receive shift register.
10. Repeat Steps 6 – 9 until all expected bytes have been received.

Transmisión de datos con opción ADB=1

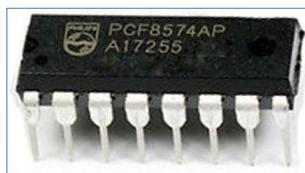
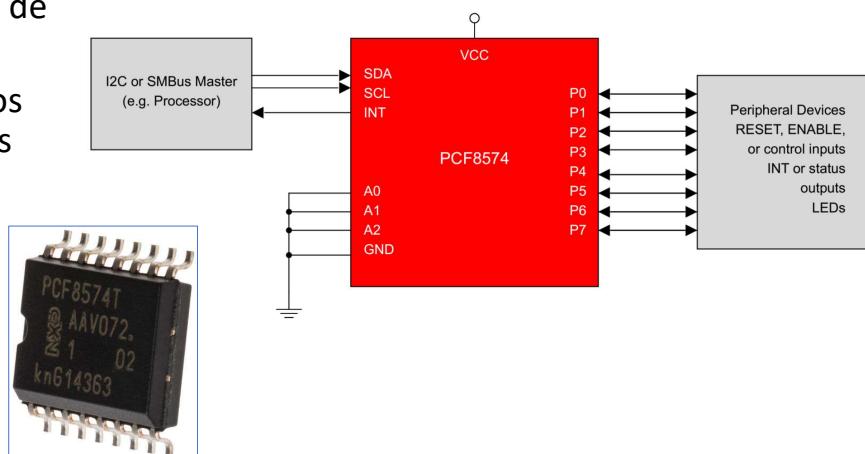


Transmisión de datos con opción ADB=0



Caso: PCF8574

- Expansor de 8 bits de E/S I2C
- Hasta 8 dispositivos PCF8574 en un bus I2C



PCF8574: Modo escritura

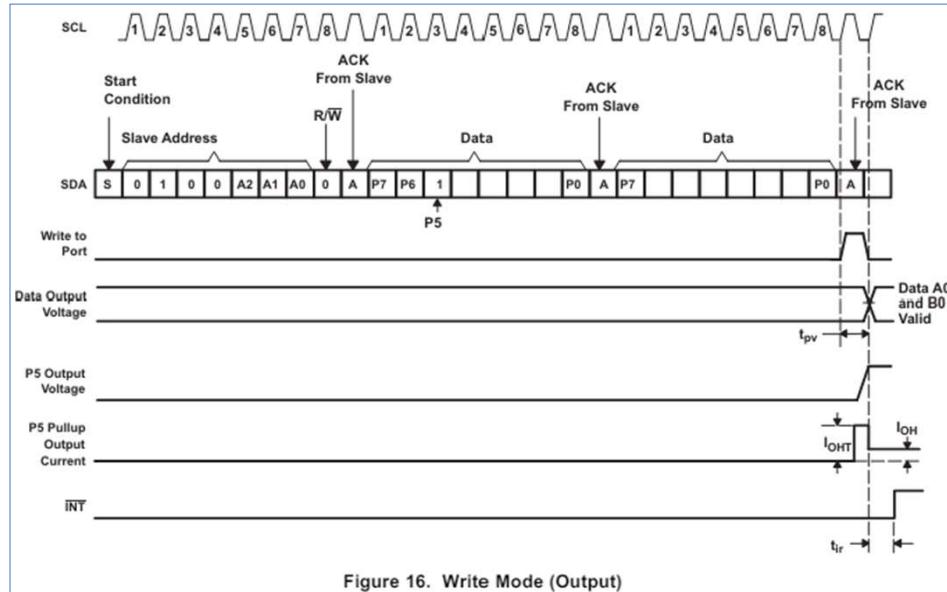


Figure 16. Write Mode (Output)

PCF8574: Modo lectura

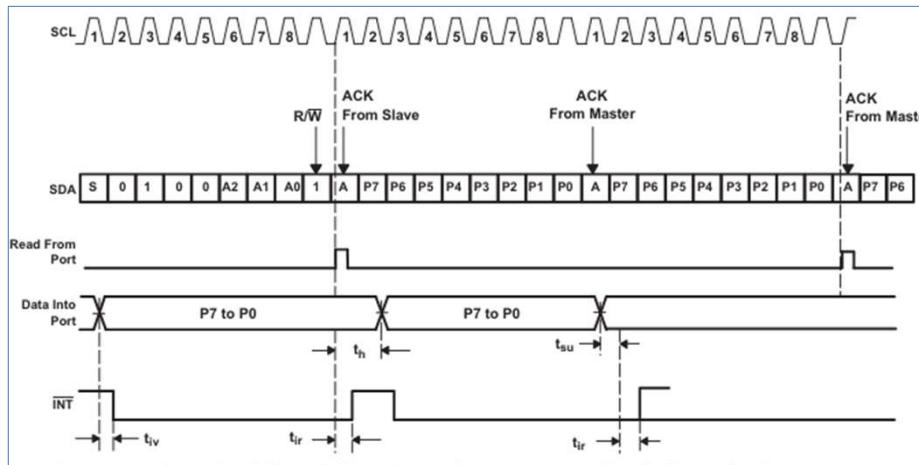


Figure 17. Read Mode (Input)

Proceso de envío de datos al PCF8574 (en I2C)

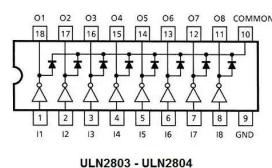
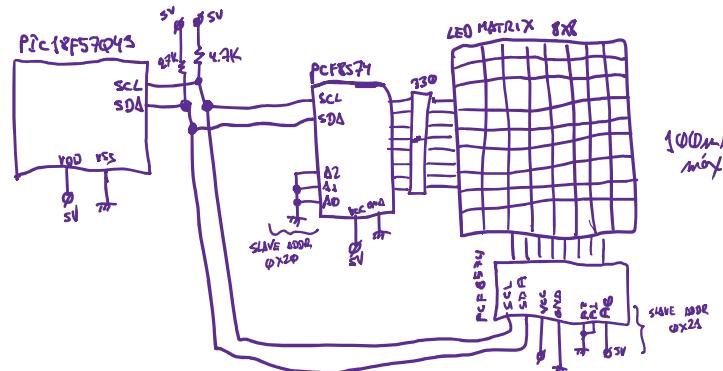
<S> <slave address + write> <ACK> <data out> <ACK> <data out> <ACK> ...
<data out> <ACK> <P>

Remark: Bold type = generated by slave device.

- Primero enviar condición de START
- Enviar la palabra de control (dirección de esclavo junto con la acción de escritura R/ \sim W = 0)
- Esperar señal de ACK de parte del esclavo
- El host envía el dato de 8 bits
- Esperar señal de ACK de parte del esclavo
- El host envía condición de STOP

Caso: PCF8574

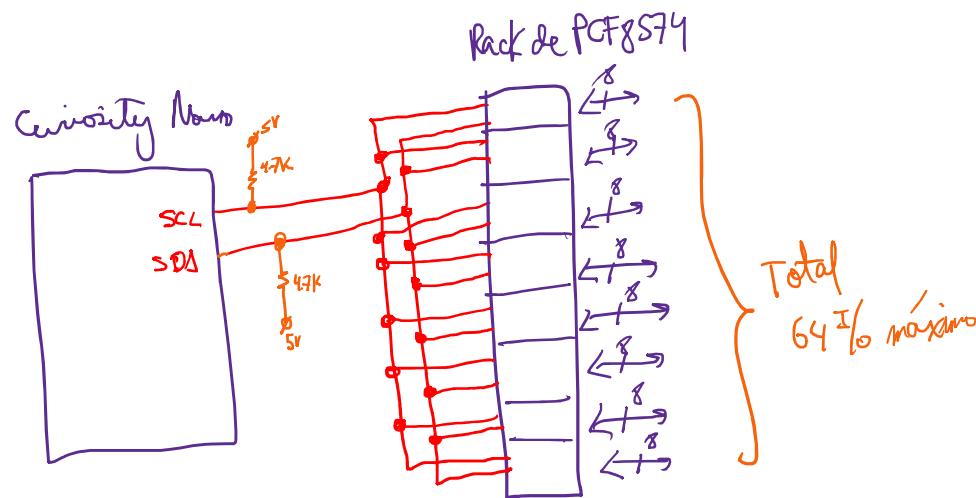
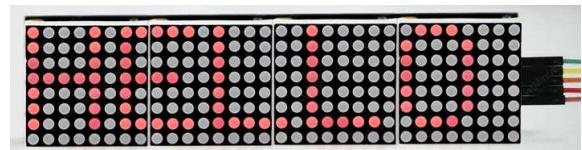
- Expansor de 8 bits de E/S I2C
- Hasta 8 dispositivos PCF8574 en un bus I2C



ROW	COL 1	2	3	4	5	6	7	8
PIN	⑤	③	④	⑩	⑥	⑪	⑫	⑯
1	①	②	⑦	⑧	⑨	⑩	⑪	⑫
2	⑩	⑪	⑫	⑬	⑭	⑮	⑯	⑰
3	⑮	⑯	⑰	⑱	⑲	⑳	⑳	⑳
4	⑰	⑱	⑲	⑳	⑳	⑳	⑳	⑳
5	⑱	⑲	⑳	⑳	⑳	⑳	⑳	⑳
6	⑲	⑳	⑳	⑳	⑳	⑳	⑳	⑳
7	⑳	⑳	⑳	⑳	⑳	⑳	⑳	⑳
8	⑳	⑳	⑳	⑳	⑳	⑳	⑳	⑳



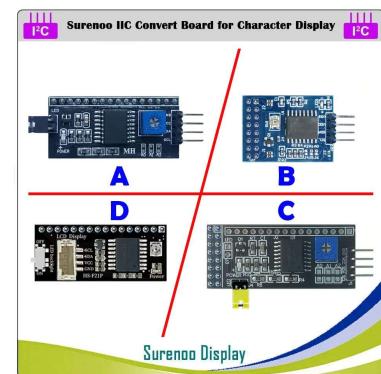
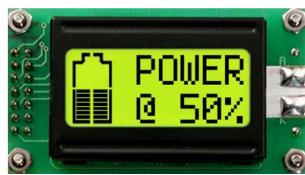
Caso: PCF8574



Caso: PCF8574



- Tarjeta I2C para display LCD 16x2 HD44780
- Posee un conector de 4 pines (Vcc, Gnd, SDA y SCL)
- Utiliza un PCF8574!

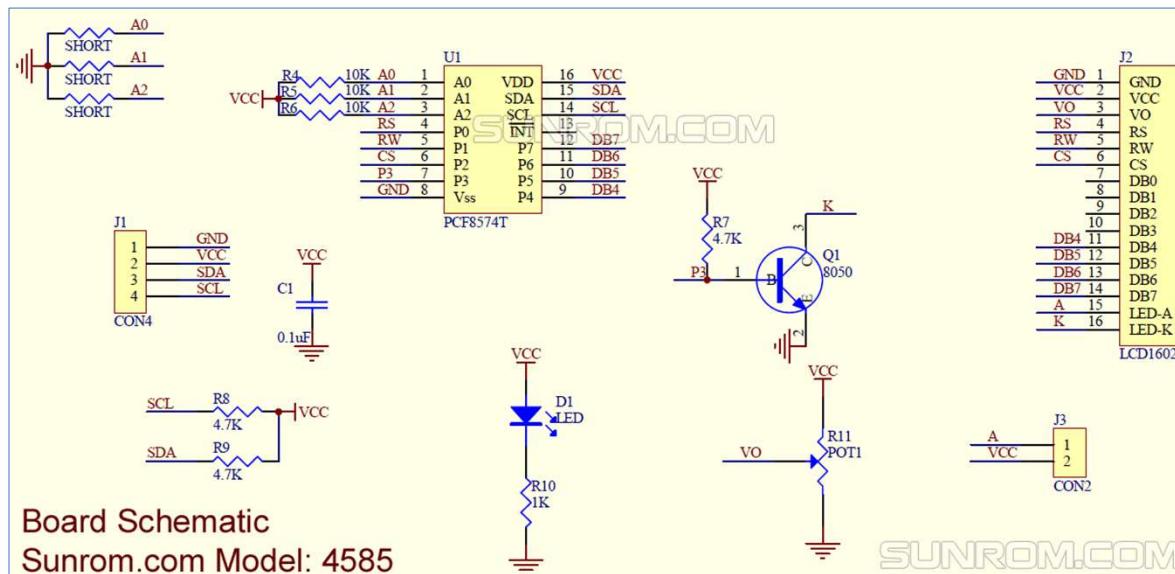


Comunicación I2C: Interface a IC PCF8574

- Por defecto la tarjeta tiene dirección de esclavo 0x27



Comunicación I2C: Interface a IC PCF8574



Librería I2C_LCD

- Se ha desarrollado una librería para manipular el LCD a través del módulo adaptador I2C.
- Revisar el repositorio, la librería tiene como nombre I2C_LCD

 Microchip-PIC18F57Q43

Some examples using the PIC18F57Q43 microcontroller and the Microchip PIC18F57Q43 Curiosity Development Board

Documentation:

- Product page: <https://www.microchip.com/en-us/product/PIC18F57Q43#>
- Datasheet: <https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/DataSheets/PIC18F27-47-57Q43-Microcontroller-Datasheet-XLP-DS40002147.pdf>
- Curiosity Board for PIC18F57Q43: <http://ww1.microchip.com/downloads/en/DeviceDoc/PIC18F57Q43-Curiosity-Nano-HW-UserGuide-DS40002186B.pdf>
- PIC18F57Q43 Curiosity Nano Hardware User Guide: <https://onlinedocs.microchip.com/pr/GUID-5D38BF5C-8481-465C4-BD08-1B8FA7289B2-en-US/2/index.html>

My contributions:

- LCD HD44780: https://github.com/tocache/Microchip-PIC18F57Q43/tree/main/EL256%202023-2%20Examples/2023-2_EL57_2_Sem13_UART.X
- I2C_LCD with PCF8574: https://github.com/tocache/Microchip-PIC18F57Q43/tree/main/EL256%202023-2%20Examples/2023-2_EL57_2_Sem14_I2C.X
- MCP23017: https://github.com/tocache/Microchip-PIC18F57Q43/tree/main/EL256%202023-2%20Examples/2023-2_EL57_2_Sem14_I2C.X

Microchip-PIC18F57Q43 / EL256 2023-2 Examples / 2023-2_EL57_2_Sem14_I2C.X / I2C_LCD.c

tocache Update I2C_LCD.c

Code Blame 473 lines (416 loc) • 14.6 KB

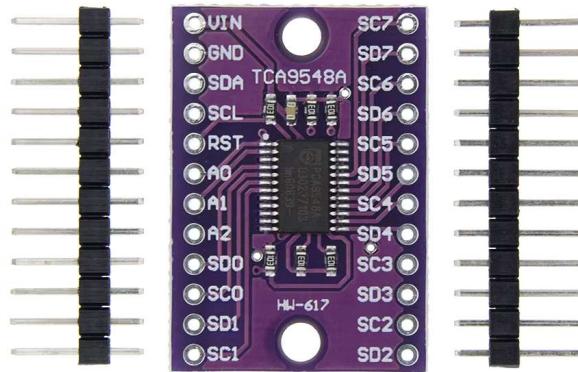
```

1 //Libreria I2C-LCD desarrollada por Kalun Lau
2 //Las principales funciones para el LCD fueron basados en una libreria
3 //desarrollada por Sergio Salas para un PIC18F4550
4 //Las configuraciones del periferico I2C1 fueron basados en un codigo
5 //proporcionado por Alonso Sanchez
6 //Las funciones de inicializacion, envio de dato y envio de comando para el LCD
7 //se basaron en el trabajo de Vladimir Anglas
8 //Curso de Microcontroladores
9 //Universidad Peruana de Ciencias Aplicadas
10 //Ultima edicion 19/11/2023
11
12 #include <xc.h>
13 #include "I2C_LCD.h"
14 #include <string.h>
15
16 void I2C1_INIT(void){
17     //configuracion del I2C
18     TRISCbts.TRISCB3 = 0; // outputs
19     TRISCbts.TRISCB4 = 0;
20     ANSELCbts.ANSELCB3 = 0; // digitales
21     ANSELCbts.ANSELCB4 = 0;
22     ODCONCbts.ODCCB3 = 1; // open drain
23     ODCONCbts.ODCCB4 = 1;

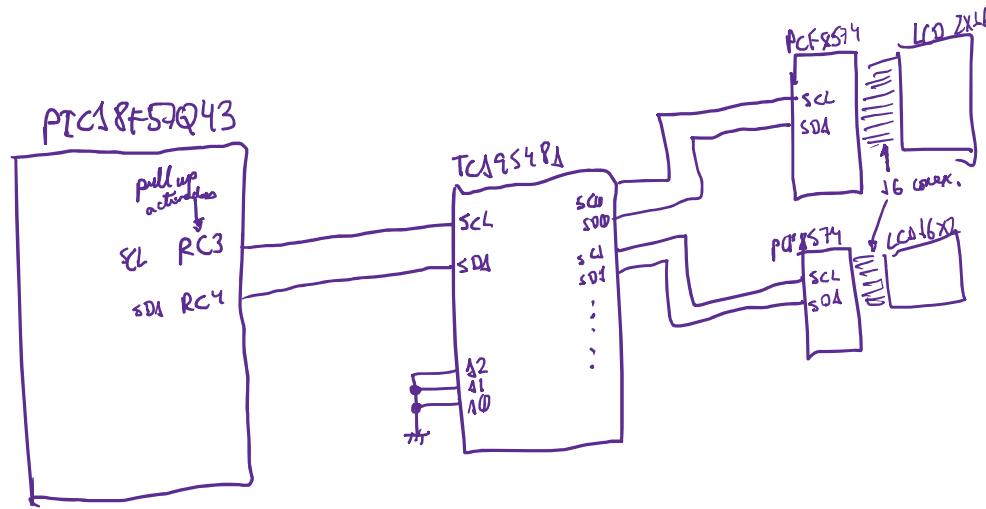
```

¿Qué hago si necesito conectar mas de 8 dispositivos PCF8574 en las líneas de I2C1 del PIC18F57Q43?

- Vamos a utilizar como alternativa de solución el dispositivo ~~PCF8575 (I/O expander de 16 bits)~~ TCA9548A (multiplexor de líneas I2C – hasta 8 líneas I2C)

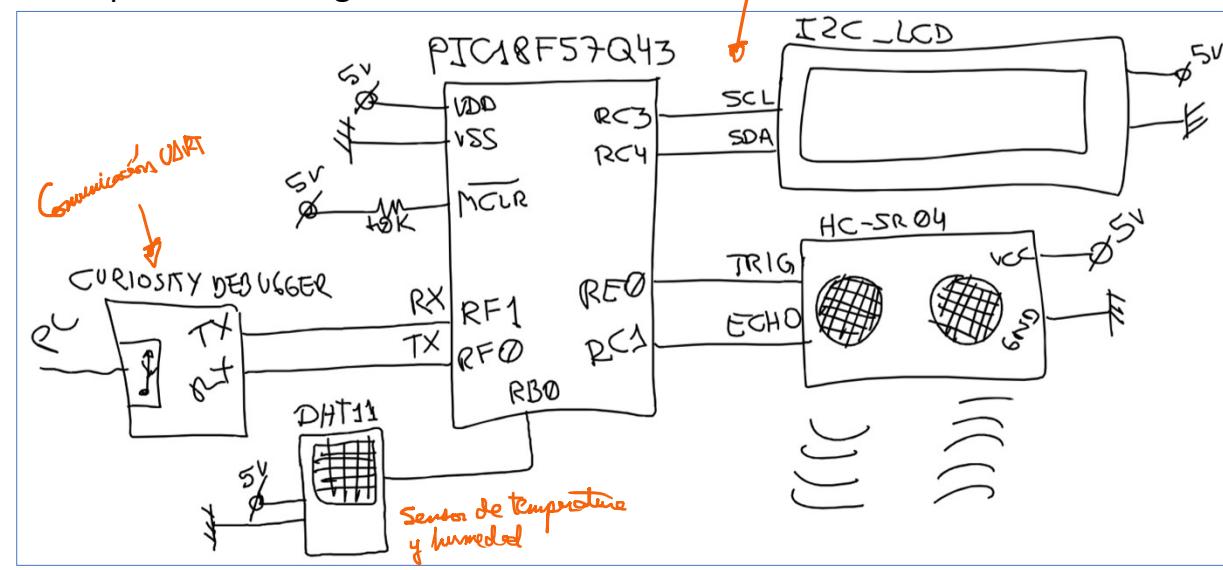


¿Cómo conecto el TCA9548A con el PIC18F57Q43?



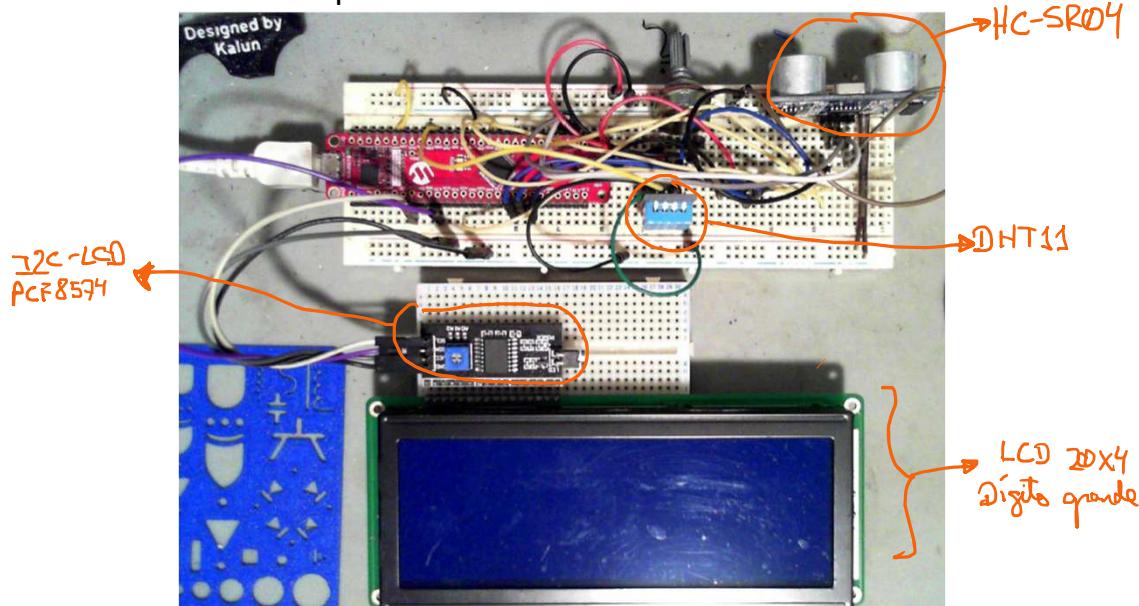
Ejemplo de aplicación 2024-1

- Implementar el siguiente circuito:



Ejemplo de aplicación 2024-1

- Circuito implementado:



Ejemplo de aplicación 2024-1

Detalle del circuito:

- Sensor DHT11 ó DHT22 conectado a RB0.
- Módulo I2C montado en el LCD 16x2 HD44780 y conectado el SCL a RC3 y SDA a RC4.
- Módulo HC-SR04 conectado el TRIG a RE0 y ECHO a RC1.
- Se empleará el puerto serial integrado en Curiosity Nano para el envío y recepción de datos hacia/desde la PC empleando un software de terminal serial (por ejemplo el Serial Monitor del Arduino IDE)

Ejemplo de aplicación 2024-1

- Las funciones para el uso del UART1 se encuentran en el repositorio:



Some examples using the PIC18F57Q43 microcontroller and the Microchip PIC18F57Q43 Curiosity Development Board

Documentation:

- Product page: <https://www.microchip.com/en-us/product/PIC18F57Q43>
- Datasheet: <https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/DataSheets/PIC18F7-47-57Q43-Microcontroller-Data-Sheet-XLP-D540002147.pdf>
- Curiosity Board for PIC18F57Q43: <http://ww1.microchip.com/downloads/en/DeviceDoc/PIC18F57Q43-Curiosity-Nano-HW-UserGuide-DS40002186B.pdf>
- PIC18F57Q43 Curiosity Nano Hardware User Guide: <https://onlinedocs.microchip.com/pr/GUID-5D38BF5C-8481-46C4-BD08-1B8F4C7289B2-en-US-2/index.html>

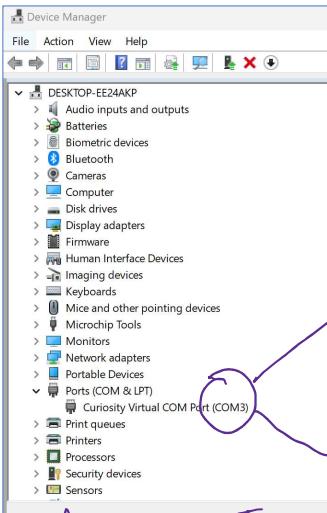
My contributions:

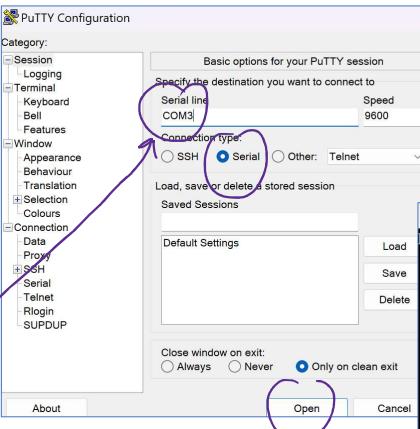
- LCD HD44780: https://github.com/tocache/Microchip-PIC18F57Q43/tree/main/EL256%202023-2%20Examples/2023-2_EL57_2_Sem13_UART_X
- I2C_LCD with PCF8574: https://github.com/tocache/Microchip-PIC18F57Q43/tree/main/EL256%202023-2%20Examples/2023-2_EL57_2_Sem14_I2C_X
- MCP23017: https://github.com/tocache/Microchip-PIC18F57Q43/tree/main/EL256%202023-2%20Examples/2023-2_EL57_2_Sem14_I2C_X
- RTC with Timer1: https://github.com/tocache/Microchip-PIC18F57Q43/blob/main/EL256%202023-2%20Examples/2023-2_EL57_2_Sem10_reloj1/maincode02.c
- UART TX & RX: https://github.com/tocache/Microchip-PIC18F57Q43/blob/main/EL256%202023-2%20Examples/2023-2_EL52_2_Sem13_UART_X/maincode05.c

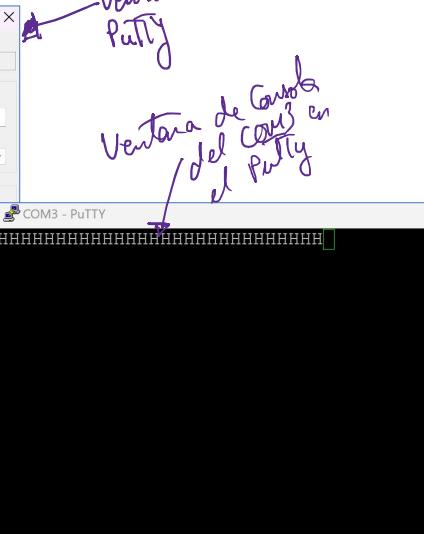
25/11/2025

Ejemplo de aplicación 2024-1

- Revisión del Puerto serial en la PC







Ventana inicial del PuTTY

Ventana de Configuración del Puerto Serial en el PuTTY

Puerto serial creado luego de conectar el Curiosity Nano

Administradores de Dispositivos del Windows

Ejemplo de aplicación 2024-1

- Pruebas de envío de datos vía UART al terminal serial de la PC

The screenshot shows the MPLAB X IDE interface. The top menu bar includes Production, Debug, Team, Tools, Window, Help, and a search bar. Below the menu is a toolbar with various icons. The main window has tabs for Start Page, MPLAB X Store, Kit Window, cabcezota.h, maincode01.c, and COM3 - PutTY. The code editor displays the following C code:

```
while(U1ERRIRbits.TXMTI
28 }
29
30 void U1_NEWLINE(void) {
31     U1_BYTE_SEND(0xA);
32     U1_BYTE_SEND(0xD);
33 }
34
35 void main(void) {
36     configuro();
37     while(1) {
38         U1_BYTE_SEND('H');
39         U1_BYTE_SEND('o');
40         U1_BYTE_SEND('l');
```

The terminal window on the right shows a continuous stream of the word "Hola" being printed.

Ejemplo de aplicación 2024-1

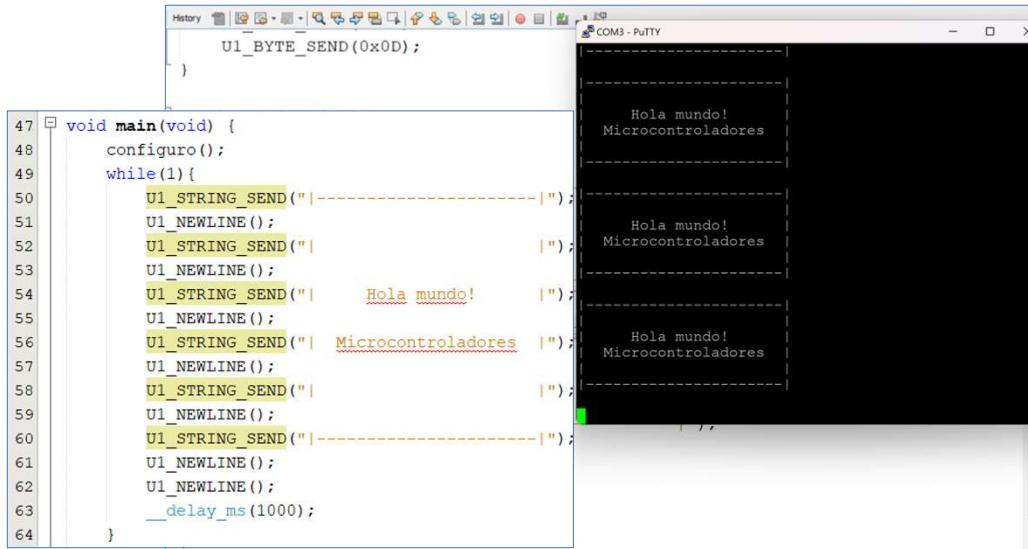
- Pruebas de envío de datos vía UART al terminal serial de la PC

The screenshot shows the MPLAB X IDE interface with the following details:

- Project Structure:** The project is named "cabcezota.h".
- Code Editor:** The main code file is "maincode01.c". It contains the following C code:40 }
41
42 void U1_NEWLINE(void){
43 U1_BYTE_SEND(0x0A);
44 U1_BYTE_SEND(0x0D);
45 }
46
47 void main(void) {
48 configuro();
49 while(1){
50 U1_STRING_SEND("Hola mundo!");
51 U1_NEWLINE();
52 __delay_ms(1000);
53 }
54 }
- Terminal Output:** The terminal window titled "COM3 - PuTTY" displays the string "Hola mundo!" repeated 15 times.

Ejemplo de aplicación 2024-1

- Pruebas de envío de datos vía UART al terminal serial de la PC



The screenshot shows a software interface with two main windows. On the left is a code editor window titled 'U1_BYTE_SEND (0x0D);' containing C-like pseudocode for a microcontroller application. The code includes several calls to 'U1_STRING_SEND' with various strings and newlines. On the right is a terminal window titled 'COM3 - PuTTY' showing the output of the application. The terminal displays three identical messages: 'Hola mundo! Microcontroladores' followed by a carriage return and a line feed.

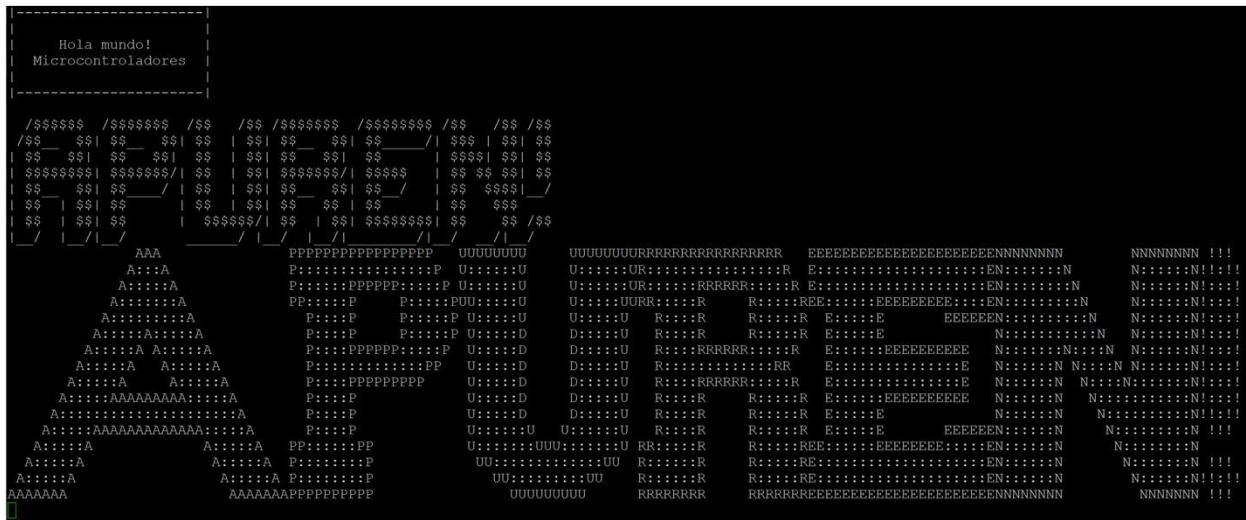
```

47 void main(void) {
48     configuro();
49     while(1){
50         U1_STRING_SEND("-----");
51         U1_NEWLINE();
52         U1_STRING_SEND("-----");
53         U1_NEWLINE();
54         U1_STRING_SEND("----- Hola mundo!");
55         U1_NEWLINE();
56         U1_STRING_SEND("----- Microcontroladores");
57         U1_NEWLINE();
58         U1_STRING_SEND("-----");
59         U1_NEWLINE();
60         U1_STRING_SEND("-----");
61         U1_NEWLINE();
62         U1_NEWLINE();
63         delay_ms(1000);
64     }
}

```

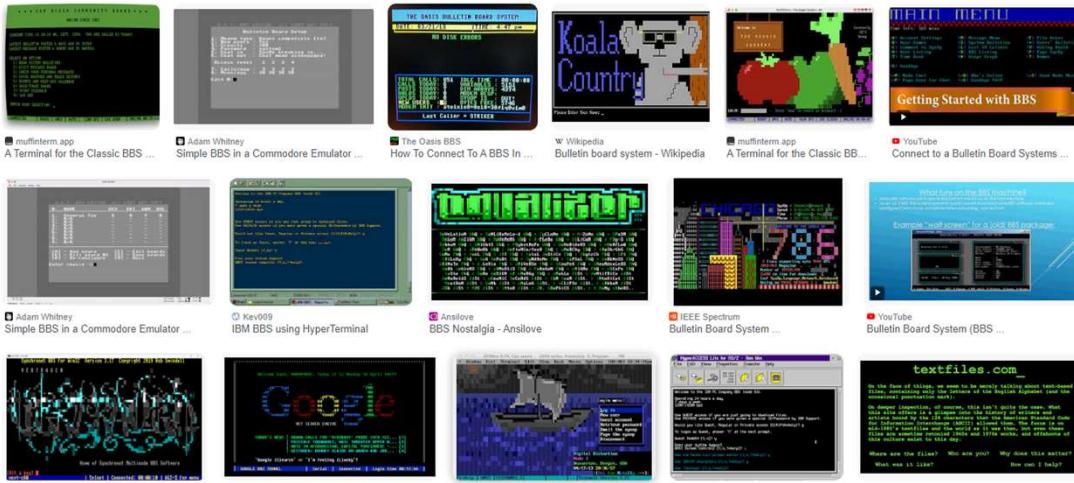
Ejemplo de aplicación 2024-1

- Pruebas de envío de datos vía UART al terminal serial de la PC
- <https://patorjk.com/software/taag/>



Ejemplo de aplicación 2024-1

- Antiguas visualizaciones en BBS



Ejemplo de aplicación 2024-1

- La librería I2C-LCD a emplear en los siguientes ejemplos se encuentra en el repositorio:
- La función `I2C_LCD_INIT()` es el empleado para inicializar el periférico I2C1 del PIC18F57Q43 y para inicializar el modulo LCD. Se deberá llamar a dicha función antes de operar el LCD.
- Las funciones para operar el display HD44780 son muy similares a las empleadas en la librería LCD vista anteriormente.

Microchip-PIC18F57Q43

Some examples using the PIC18F57Q43 microcontroller and the Microchip Pic18F57Q43 Curiosity Development Board

Documentation:
-Product page: <https://www.microchip.com/en-us/product/PIC18F57Q43#>
-Datasheet: <https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/DataSheets/PIC18F27-47-57Q43-Microcontroller-Datasheet-XLP-DS40002147.pdf>
-Curiosity Board for PIC18F57Q43: <http://ww1.microchip.com/downloads/en/DeviceDoc/PIC18F57Q43-Curiosity-Nano-HW-UserGuide-DS400021868.pdf>

My contributions:
-LCD HD44780: https://github.com/tocache/Microchip-PIC18F57Q43/tree/main/EL256%202023-2%20Examples/2023-2_EL57_2_Sem13_UART_X
-I2C_LCD with PCF8574: https://github.com/tocache/Microchip-PIC18F57Q43/tree/main/EL256%202023-2%20Examples/2023-2_EL57_2_Sem14_I2CX
-MCP23017: https://github.com/tocache/Microchip-PIC18F57Q43/tree/main/EL256%202023-2%20Examples/2023-2_EL57_2_Sem14_I2CX
-RTC with Timer1: https://github.com/tocache/Microchip-PIC18F57Q43/blob/main/EL256%202023-2%20Examples/2023-2_EL57_2_Sem10_relojX/maincode02.c
-UART TX & RX: https://github.com/tocache/Microchip-PIC18F57Q43/blob/main/EL256%202023-2%20Examples/2023-2_EL52_2_Sem13_UART_X/maincode05.c
-I2C_SDO4_usina_ESM: https://github.com/tocache/Microchip_PIC18F57Q43/tree/main/EL256%202023-2%20Examples/2023-2_EL52_2_Sem13_I2CSDO4_usina_ESM

Ejemplo de aplicación 2024-1

- Evidencia de visualización en el LCD 16x2 conectado con el I2C-LCD

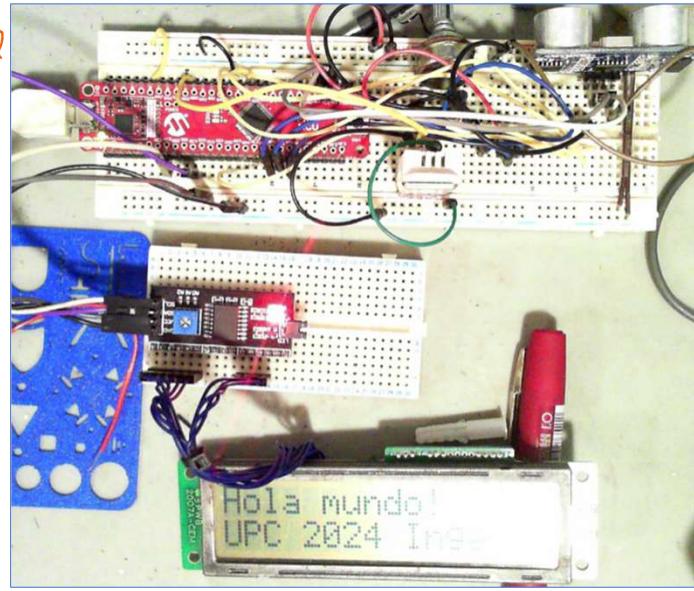
```

48 void main(void) {
49     configuro();
50     I2C_LCD_INIT();
51     while(1){
52         UI_STRING_SEND("-----");
53         UI_NEWLINE();
54         UI_STRING_SEND("      |");
55         UI_NEWLINE();
56         UI_STRING_SEND("      | Hola mundo!");
57         UI_NEWLINE();
58         UI_STRING_SEND("      | Microcontroladores");
59         UI_NEWLINE();
60         UI_STRING_SEND("      |");
61         UI_NEWLINE();
62         UI_STRING_SEND("-----");
63         UI_NEWLINE();
64         UI_NEWLINE();
65         I2C_POS_CURSOR(1,0);
66         I2C_ESCRIBE_MENSAJE2("Hola mundo!");
67         I2C_POS_CURSOR(2,0);
68         I2C_ESCRIBE_MENSAJE2("UPC 2024 Ing.");
69         _delay_ms(1000);
70     }
71 }
```

Configuración del PCF8574 del I2C-LCD y del LCD en modo 16x2

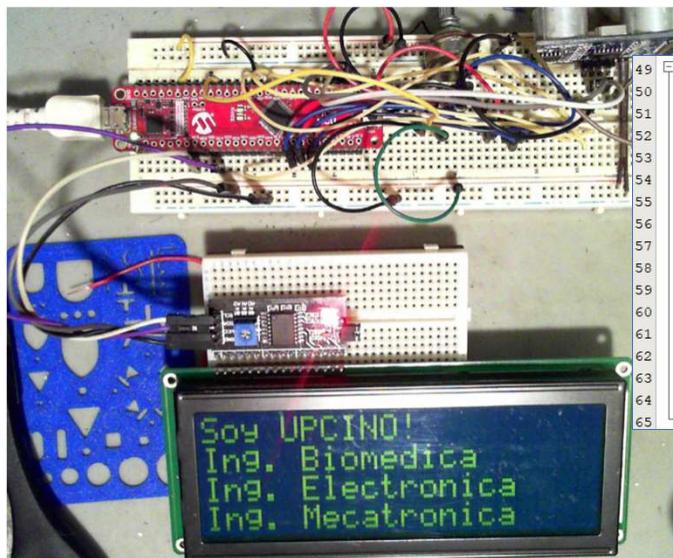
Comunicación Serial

control I2C



Ejemplo de aplicación 2024-1

- Pruebas de impresión en LCD con cuatro líneas:

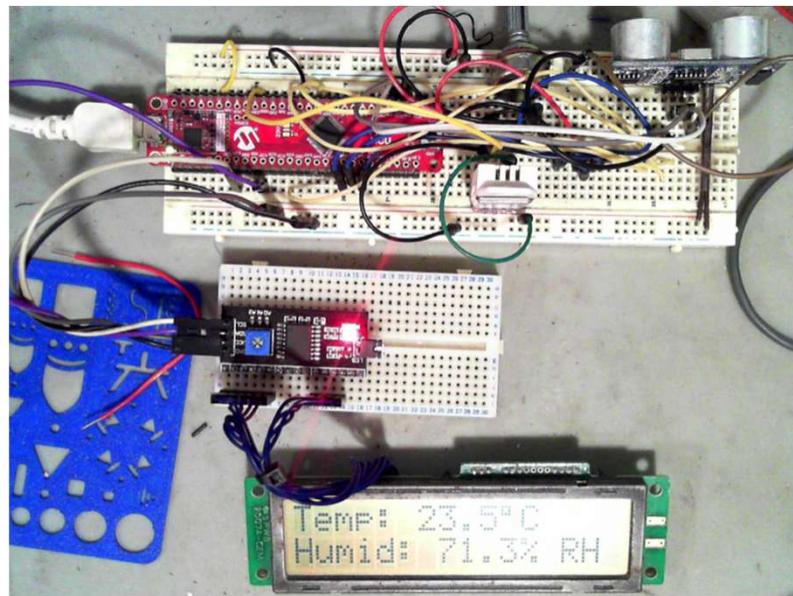


```

49 void main(void) {
50     configuro();
51     I2C_LCD_INIT();
52     while(1{
53         UI_STRING_SEND("Hola mundo! Mi nombre es Kalun");
54         UI_NEWLINE();
55         I2C_POS_CURSOR(1,0);
56         I2C_ESCRIBE_MENSAJE2("Soy UPCINO!");
57         I2C_POS_CURSOR(2,0);
58         I2C_ESCRIBE_MENSAJE2("Ing. Biomedica");
59         I2C_POS_CURSOR(3,0);
60         I2C_ESCRIBE_MENSAJE2("Ing. Electronica");
61         I2C_POS_CURSOR(4,0);
62         I2C_ESCRIBE_MENSAJE2("Ing. Mecatronica");
63         _delay_ms(2000);
64     }
65 }
```

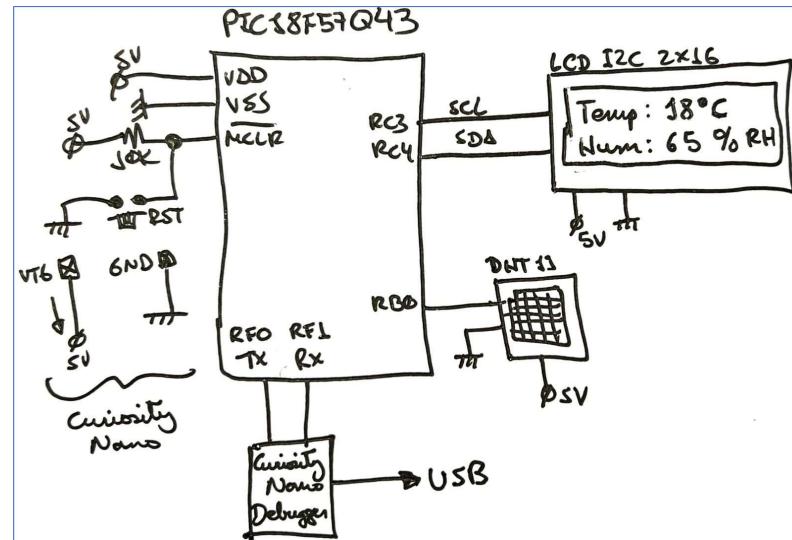
Ejemplo de aplicación 2024-1

- Pruebas de medición con el sensor DHT22:



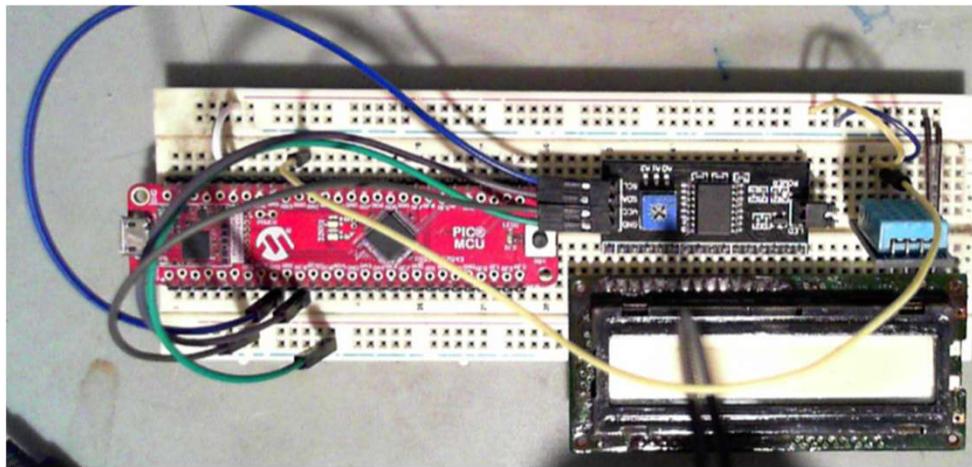
Ejemplo 2024-2

- El LCD 2x16 tiene montado el módulo I2C-LCD, para ello vamos a emplear la librería I2C_LCD.
- El DHT11 esta montado en el puerto RBO y se empleará la librería LIB_DHT.
- Para la comunicación serial UART se empleará el U1, con velocidad y protocolo 9600 8N1. Se empleará la librería LIB_UART.



Ejemplo 2024-2

- Implementación del circuito de pruebas:



Ejemplo 2024-2

- Librerías en el repositorio en Github del profesor Kalun:

Name	Last commit message
I2C_LCD.c	Add files via upload
I2C_LCD.h	Add files via upload
LCD.c	Update LCD.c
LCD.h	Add files via upload
LIB_DHT.c	Add files via upload
LIB_DHT.h	Add files via upload
LIB_UART.c	Add files via upload
LIB_UART.h	Add files via upload

Ejemplo 2024-2

- Código ejemplo para visualizar mensajes en el I2C_LCD:

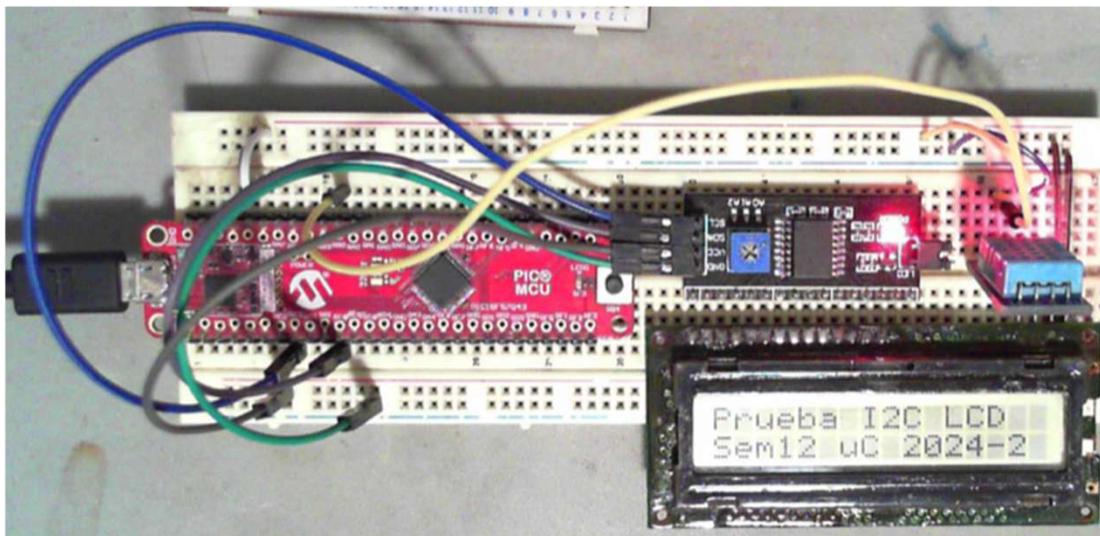
```

MPLAB X IDE v6.20 - ELS1_1_Sem12_I2CUARTDH11 : default
File Edit View Navigate Source Refactor Production Debug Team Tools Window Help
Projects x Files default
EL51_1_Sem12_I2CUARTDH11
  Header Files
    cabecera.h
    I2C_LCD.h
  Important Files
  Source Files
    I2C_LCD.c
    mancode01.c
  Libraries
  Loadables
Source History Kit Window Start Page MPLAB X Store cabecera.h mancode01.c I2C_LCD.c
1 /* ... 6 lines */
2
3 #include <xc.h>
4 #include "cabecera.h"
5 #include "I2C_LCD.h"
6 #define _XTAL_FREQ 32000000UL
7
8 void configuró(void){
9     OSCCON1 = 0x60;
10    OSCFRQ = 0x06;
11    OSCEN = 0x40;
12    I2C_LCD_INIT();
13 }
14
15 void main(void) {
16     configuró();
17     I2C_POS_CURSOR(1,0);
18     I2C_ESCRIBE_MENSAJE2("Prueba I2C LCD");
19     I2C_POS_CURSOR(2,0);
20     I2C_ESCRIBE_MENSAJE2("Sem12 uC 2024-2");
21     while(1){
22     }
23 }
24
25
26
27
28

```

Ejemplo 2024-2

- Pruebas de visualización en el I2C_LCD:



Ejemplo 2024-2

- Pruebas de envío de datos al puerto serial por U1

```

#include <xc.h>
#include "cabecera.h"
#include "I2C_LCD.h"
#include "LIB_UART.h"

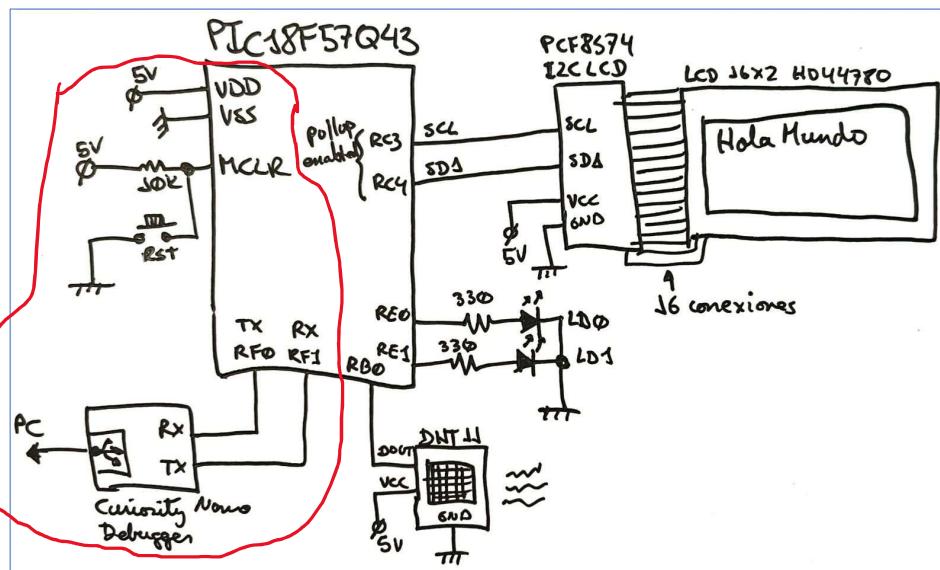
void configuracion(void) {
    OSCCON1 = 0x60;
    OSCFRC = 0x06;
    OSCEN = 0x40;
    I2C_LCD_INIT();
    U1_INIT(BAUD_9600);
}

void main(void) {
    configuracion();
    I2C_POS_CURSOR(1,0);
    I2C_ESCRIBE_MENSAJE2("Prueba I2C LCD");
    I2C_POS_CURSOR(2,0);
    I2C_ESCRIBE_MENSAJE2("Semana 12 uC 2024-2");
    U1_STRING_SEND("Prueba comunicacion UART con U1 a 9600 b");
    U1_NEWLINE();
    U1_STRING_SEND("Semana 12 Microcontroladores 2024-2");
    U1_NEWLINE();
    while(1);
}

```

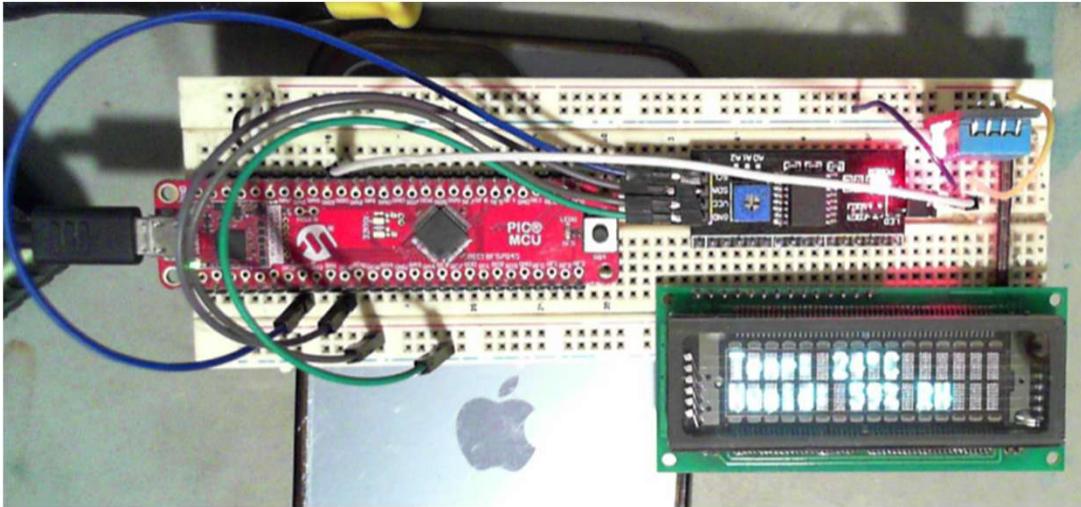
Asignación Semana 13 2024-2

- Interacción con el PIC18F57Q43 mediante el puerto UART



Asignación Semana 13 2024-2

- Implementación del prototipo



Asignación Semana 13 2024-2

- Plantilla inicial:

MPLAB X IDE v6.20 - ELS1_1_Sem13_UART_RXTX : default

Projects > Files > ELS1_1_Sem13_UART_RXTX > maincode01.c

```

7 #include <xc.h>
8 #include "I2C_LCD.h"
9 #include "LIB_UART.h"
10 #include "LIB_DHT.h"
11 #define _XTAL_FREQ 32000000UL
12
13 void configuro(void){
14     //conf de la fuente de reloj
15     OSCCON1 = 0x60;
16     OSCFRQ = 0x06;
17     OSCEN = 0x40;
18     //conf de las E/S
19     TRISEbits.TRISE0 = 0;    //RE0 como salida
20     TRISEbits.TRISE1 = 0;    //RE1 como salida
21     ANSELEbits.ANSELE0 = 0; //RE0 como digital
22     ANSELEbits.ANSELE1 = 0; //RE1 como digital
23     //inicialización del I2CLCD
24     I2C_LCD_INIT();
25     //inicialización del UART
26     U1_INIT(BAUD_9600);
27 }
28
29 void main(void) {
30     configuro();
31 }
```

falta: #include "cabecera.h"

Asignación Semana 13 2024-2

- Objetivos:

- Tomar lectura del valor de temperatura y humedad del DHT11 y visualizarlo en el I2CLCD

```

40 void main(void) {
41     configuro();
42     pantallon();
43     while(1){
44         temperatura = DHT_GetTemp(DHT11);
45         __delay_ms(500);
46         humedad = DHT_GetHumid(DHT11);
47         __delay_ms(500);
48         I2C_POS_CURSOR(1,0);
49         I2C_ESCRIBE_MENSAJE2("Temp: ");
50         I2C_LCD_ESCRIBE_VAR_INT(temperatura, 2, 0);
51         //imprimir la unidad de medida de grado centigrado
52         I2C_ENVIA_LCD_DATA(0xDF);    //simbolo de grado
53         I2C_ENVIA_LCD_DATA("C");
54         I2C_POS_CURSOR(2,0);
55         I2C_ESCRIBE_MENSAJE2("Hume:");
56         I2C_LCD_ESCRIBE_VAR_INT(humedad, 2, 0);
57         //imprimir unidad de medida de humedad
58         I2C_ESCRIBE_MENSAJE2("%RH");
59     }
60 }
```

DHT11 I2C LCD Variables globales unsigned int (16 bits)

Asignación Semana 13 2024-2

- Análisis:

- Diferencia entre los niveles de optimización del XC8 (entre L0 y Ls)

Level 0		Level s	
18F57Q43 Memory Summary:		18F57Q43 Memory Summary:	
Program space	used 13BDh (5053) of 20000h bytes (3.9%)	Program space	used EF2h (3826) of 20000h bytes (2.9%)
Data space	used 1Ch (28) of 2000h bytes (0.3%)	Data space	used 1Bh (27) of 2000h bytes (0.3%)
Configuration bits	used 0h (0) of 9h words (0.0%)	Configuration bits	used 0h (0) of 9h words (0.0%)
EEPROM space	used 0h (0) of 400h bytes (0.0%)	EEPROM space	used 0h (0) of 400h bytes (0.0%)
ID Location space	used 0h (0) of 40h bytes (0.0%)	ID Location space	used 0h (0) of 40h bytes (0.0%)

Asignación Semana 13 2024-2

- Objetivos:

- Tomar lectura del valor de temperatura y humedad del DHT11, visualizarlo en el I2CLCD y en el terminal serial.

```

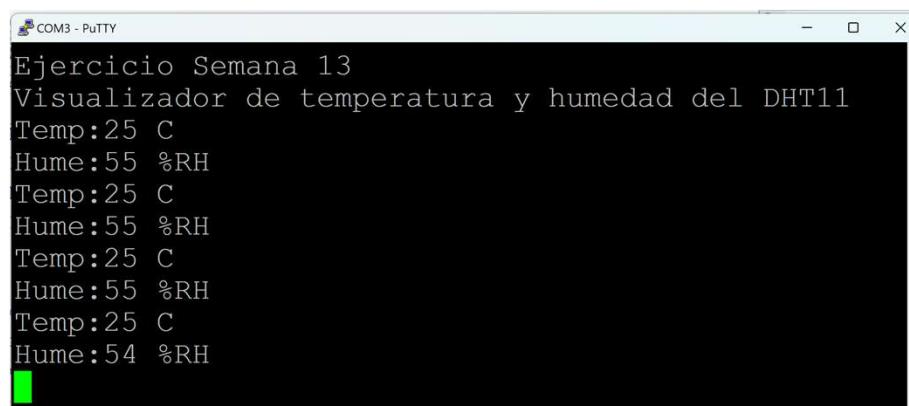
68  void vis_uart(void){
69      U1_STRING_SEND("Temp:");
70      U1_VAR_INT(temperatura, 2, 0);
71      U1_STRING_SEND(" C");
72      U1_NEWLINE();
73      U1_STRING_SEND("Hume:");
74      U1_VAR_INT(humedad, 2, 0);
75      U1_STRING_SEND(" %RH");
76      U1_NEWLINE();
77  }
78
79  void main(void) {
80      configuro();
81      pantalla();
82      while(1){
83          temperatura = DHT_GetTemp(DHT11);
84          _delay_ms(1000);
85          humedad = DHT_GetHumid(DHT11);
86          _delay_ms(1000);
87          vis_i2c_lcd();
88          vis_uart();
89      }
90  }

```

Asignación Semana 13 2024-2

- Objetivos:

- Tomar lectura del valor de temperatura y humedad del DHT11, visualizarlo en el I2CLCD y en el terminal serial.



Asignación Semana 13 2024-2

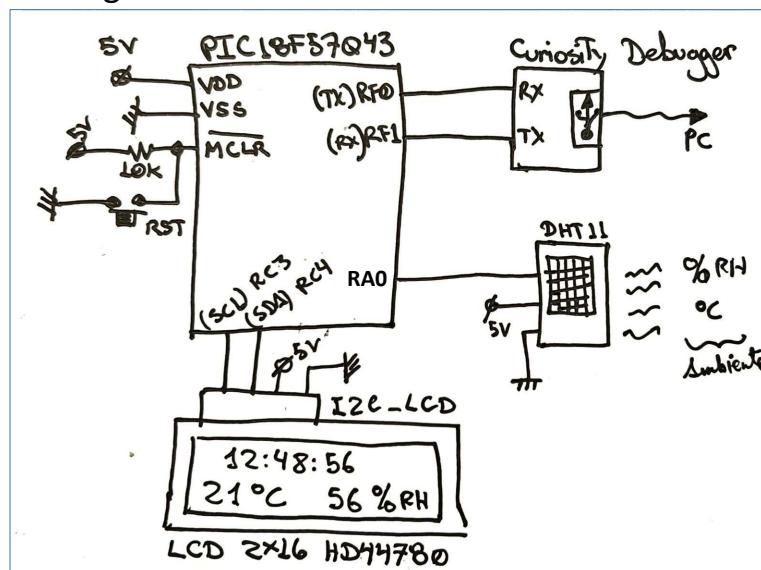
- Asignación:

- Establecer umbrales y visualizarlos en los LEDs
 - Cuando la temperatura supere los 27°C deberá de encender LD0
 - Cuando la humedad supere los 70%RH deberá de encender LD1
- Con el pulsador integrado (RB4 en interrupción externa) deberán de intercambiar la visualización de la temperatura, entre °C y °F.

Pasar de grados Celsius (°C)	Pasar de grados Fahrenheit (°F)
a Fahrenheit (°F)	a Celsius (°C)
${}^{\circ}\text{F} = ({}^{\circ}\text{C} \cdot 1,8) + 32$	${}^{\circ}\text{C} = ({}^{\circ}\text{F} - 32) / 1,8$

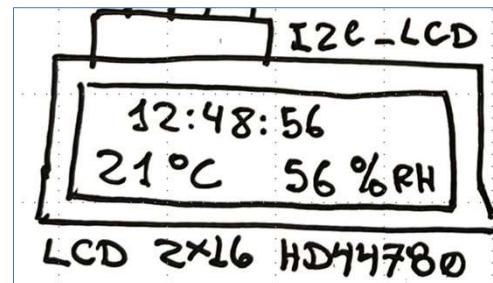
Ejercicio 2025-2 Semana 13

- Implementar el siguiente circuito:

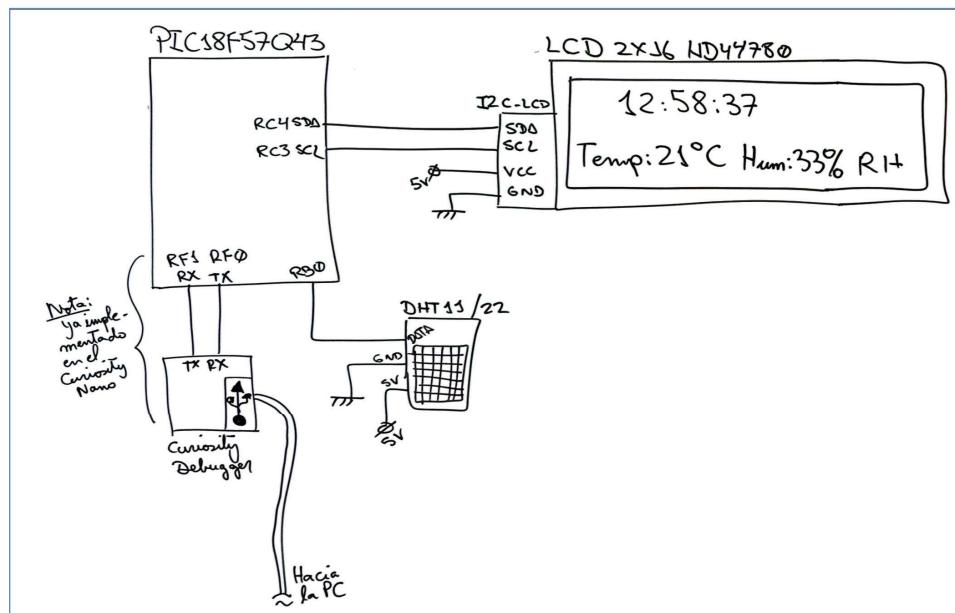


Minilab Semana 13 2025-2

- Agregar funcionalidad de la hora visto en semana 11
- El ajuste de la hora actual será a través de la comunicación UART, desde el terminal serial de la PC se deberá de enviar los ajustes de horas y minutos al Curiosity Nano PIC18F57Q43 a una velocidad de 9600 8N1.
- Hora de término 21:00



Implementar el siguiente circuito



```

7   #include <xc.h>
8   #include "cabecera.h"
9   #include "I2C_LCD.h"
10  #include "LIB_DHT.h"
11  #include "LIB_UART.h"
12  #define _XTAL_FREQ 32000000UL
13
14  void configuro(void){
15      //con modulo oscilador
16      OSCCON1 = 0x60;
17      OSCFRQ = 0x06;
18      OSCEN = 0x40;
19      //inicializacion de librerias
20      I2C_LCD_INIT();
21      U1_INIT(BAUD_9600);
22  }
23
24  void splash_screen(void){
25      I2C_POS_CURSOR(1,0);
26      I2C_ESCRIBE_MENSAJE2("Hola mundo PAPU");
27      I2C_POS_CURSOR(2,0);
28      I2C_ESCRIBE_MENSAJE2("Microbios 20252");
29      I2C_POS_CURSOR(2,0);
30      unsigned char x_var;
31      for(x_var=0;x_var<16;x_var++){
32          I2C_ENVIA_LCD_DATA(' ');
33          __delay_ms(100);
34      }
35      __delay_ms(3000);
36      I2C_BORRAR_LCD();
37  }
39  unsigned int cuenta=0;
40
41  void main(void) {
42      configuro();
43      splash_screen();
44      while(1{
45          U1_STRING_SEND("*****");
46          U1_NEWLINE();
47          U1_STRING_SEND("* HOLA PAPU UPCINO *");
48          U1_NEWLINE();
49          U1_STRING_SEND("* PROFE AYUDE PEEE *");
50          U1_NEWLINE();
51          U1_STRING_SEND("* VEZ: ");
52          U1_VAR_INT(cuenta,5,0);
53          U1_STRING_SEND("      ");
54          U1_NEWLINE();
55          U1_STRING_SEND("*****");
56          U1_NEWLINE();
57          cuenta++;
58      }
59      __delay_ms(2000);
60  }

```

```

7   #include <xc.h>
8   #include "cabecera.h"
9   #include "I2C_LCD.h"
10  #include "LIB_DHT.h"
11  #include "LIB_UART.h"
12  #define _XTAL_FREQ 32000000UL
13
14  void configuro(void){
15      //con modulo oscilador
16      OSCCON1 = 0x60;
17      OSCFRQ = 0x06;
18      OSCEN = 0x40;
19      //inicializacion de librerias
20      I2C_LCD_INIT();
21      U1_INIT(BAUD_9600);
22  }
23
24  void splash_screen(void){
25      I2C_POS_CURSOR(1,0);
26      I2C_ESCRIBE_MENSAJE2("Hola mundo PAPU");
27      I2C_POS_CURSOR(2,0);
28      I2C_ESCRIBE_MENSAJE2("Microbios 20252");
29      I2C_POS_CURSOR(2,0);
30      unsigned char x_var;
31      for(x_var=0;x_var<16;x_var++){
32          I2C_ENVIA_LCD_DATA(' ');
33          __delay_ms(100);
34      }
35      __delay_ms(3000);
36      I2C_BORRAR_LCD();
37  }
39  unsigned int cuenta=0;
40
41  void main(void) {
42      configuro();
43      splash_screen();
44      while(1{
45          struct DHT_Values medidas = DHT_GetBoth(DHT11);
46          I2C_POS_CURSOR(1,0);
47          I2C_ESCRIBE_MENSAJE2("Temp:");
48          I2C_LCD_ESCRIBE_VAR_INT(meidas.DHT_Temp, 2, 0);
49          I2C_LCD_CHAR_GRADO();
50          I2C_ESCRIBE_MENSAJE2("C ");
51          I2C_POS_CURSOR(2,0);
52          I2C_ESCRIBE_MENSAJE2("Humed:");
53          I2C_LCD_ESCRIBE_VAR_INT(meidas.DHT_Humid, 2, 0);
54          I2C_ESCRIBE_MENSAJE2("%RH ");
55          U1_STRING_SEND("*****");
56          U1_NEWLINE();
57          U1_STRING_SEND("* HOLA PAPU UPCINO *");
58          U1_NEWLINE();
59          U1_STRING_SEND("* PROFE AYUDE PEEE *");
60          U1_NEWLINE();
61          U1_STRING_SEND("* TEMP: ");
62          U1_VAR_INT(meidas.DHT_Temp,2,0);
63          U1_STRING_SEND(" *C *");
64          U1_VAR_INT(cuenta,5,0);
65          U1_STRING_SEND("      ");
66          U1_NEWLINE();
67          U1_STRING_SEND("*****");
68          U1_VAR_INT(meidas.DHT_Humid,2,0);
69          U1_STRING_SEND(" *HUM: *");
70          U1_NEWLINE();
71          U1_STRING_SEND(" * RH *");
72          U1_VAR_INT(meidas.DHT_Humid,2,0);
73          U1_STRING_SEND("      ");
74          U1_NEWLINE();
75          U1_STRING_SEND("*****");
76          U1_NEWLINE();
77          U1_NEWLINE();
78          cuenta++;
79          __delay_ms(2000);
80      }
81  }

```

Monitor serial con el PuTTY

```
* PROFE AYUDE PEEE *
* *
* VEZ: 01625 *
* TEMP: 22 *C *
* HUM: 50 %RH *
*****
*****
* HOLA PAPU UPCINO *
* PROFE AYUDE PEEE *
* *
* VEZ: 01626 *
* TEMP: 22 *C *
* HUM: 49 %RH *
*****
*****
* HOLA PAPU UPCINO *
* PROFE AYUDE PEEE *
* *
* VEZ: 01627 *
* TEMP: 22 *C *
* HUM: 49 %RH *
*****
```

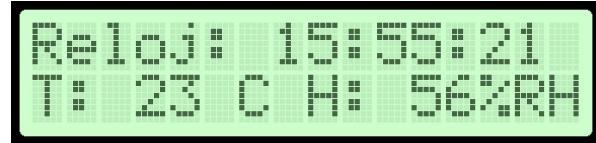
Minilab Semana 13 Turno 14:00

- Incluir el funcionamiento del reloj hecho en semana 11 en la aplicación desarrollada en esta sesión
- Carga de video en la actividad del AV Unidad 4 Semana 13:
 - Presentándote
 - Mostrando el código en XC8 C en la pantalla del MPLABX de tu PC
 - Evidencia de funcionamiento del circuito hecho en el protoboard
- Recordar que este minilab representa 2p del DD
- Límite 17:00

Reloj: 15:55:21
T: 25 °C H: 56%RH

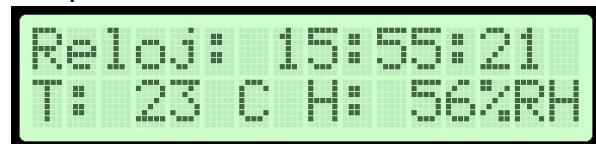
Minilab Semana 13 Turno 18:00

- Incluir el funcionamiento del reloj hecho en semana 11 en la aplicación desarrollada en esta sesión
- Carga de video en la actividad del AV Unidad 4 Semana 13:
 - Presentándose
 - Mostrando el código en XC8 C en la pantalla del MPLABX de tu PC
 - Evidencia de funcionamiento del circuito hecho en el protoboard
- Recordar que este minilab representa 2p del DD
- Límite 21:30



Minilab Semana 13 Turno 09:00

- Incluir el funcionamiento del reloj hecho en semana 11 en la aplicación desarrollada en esta sesión
- Carga de video en la actividad del AV Unidad 4 Semana 13:
 - Presentándose
 - Mostrando el código en XC8 C en la pantalla del MPLABX de tu PC
 - Evidencia de funcionamiento del circuito hecho en el protoboard
- Recordar que este minilab representa 2p del DD
- Límite 12:00



Minilab Semana 13 Turno 15:00

- Incluir el funcionamiento del reloj hecho en semana 11 en la aplicación desarrollada en esta sesión
- Carga de video en la actividad del AV Unidad 4 Semana 13:
 - Presentándose
 - Mostrando el código en XC8 C en la pantalla del MPLABX de tu PC
 - Evidencia de funcionamiento del circuito hecho en el protoboard
- Recordar que este minilab representa 2p del DD
- Límite 18:30



Reloj: 15:55:21
T: 23°C H: 56%RH

Fin de la sesión