CLASE5: USART

1. DESCRIPCION Y MANIPULACION DE FUNCIONES PARA UART
   1.1. DEFINICION

receptor transmisor asíncrono síncrono universal, es una forma de comunicación entre dispositivos que tengan esta capacidad, donde los datos son enviados en grupos de 8 bits o de 9 bits pero bit por bit, esto es en serie, por eso se dice que esta es una comunicación serial, en esta sección se comentará sobre este tipo de comunicación utilizando el módulo USART del microcontrolador PIC, con el módulo USART PIC el microcontrolador puede comunicarse e intercambiar datos con el ordenador, con otros microcontroladores, etc.

Para la comunicación entre microcontroladores y para la comunicación entre el microcontrolador y el ordenador, se necesitan 2 hilos de conducción para la transmisión y recepción de datos, y un hilo de conducción para la conexión de los comunes o GND

Si la comunicación USART PIC es asíncrona, uno de los hilos será para la transmisión de los datos de un dispositivo a otro y el otro hilo será para la recepción de datos entre un dispositivo a otro, la transmisión y la recepción pueden ocurrir en forma simultánea, lo que si se tiene que cumplir es que la frecuencia de trabajo de ambos dispositivos tiene que ser la misma, a esto se le conoce como los baudios que viene a ser la cantidad de bits por segundo que se transmitirán entre ambos dispositivos.

Si la comunicación USART PIC es síncrona, uno de los hilos será utilizado tanto para la transmisión y la recepción de datos por lo que la transmisión no puede ocurrir en forma simultánea, el otro hilo será utilizado para enviar la señal de reloj de sincronización entre ambos dispositivos, en este caso uno de los dispositivos es llamado maestro y el otro esclavo, el maestro es el que controla la señal de reloj y cuando se inicia o finaliza la comunicación.

   1.2. MANIPULACION DE DIRECTIVAS PARA EL UART

Para utilizar el UART en el PIC se debe declarar la siguiente directiva

```
#use rs232(BAUD = 9600, XMIT = PIN_C6,RCV = PIN_C7)
```

Donde:

- BAUD: Velocidad de transmisión de datos, esta velocidad debe ser la misma con el dispositivo que se desea comunicar para que la transferencia de datos sea exitosa
- XMIT: Es el pin de transmisión de datos para el PIC16F877A es el PIN_C6
- RCV: Es el pin de recepción de datos para el PIC16F877A es el pin_C7

Para transmitir datos del PIC hacia otro dispositivo se utiliza la función:

```
printf("Hola Mundo");
```

Para la recepción de datos de utiliza la función:

```
dato_recibido = getc();
```

Para utilizar la recepción de datos por interrupción se utilizan las siguientes funciones

```
enable_interrupts(INT_RDA);
enable_interrupts(GLOBAL);
```

El vector de interrupción

```
#int_RDA
void RDA_isr() {
   dato_recibido  = getc();
}
```

Utilizando las funciones del PIC también es posible generar una comunicación serial por software para llevar acabo esto se configura a las funciones de la siguiente manera:

Configuración de la directiva:

```
#use rs232(stream=swit1,baud=9600,parity=N,xmit=PIN_B0,rcv=PIN_B1,bits=8)
```

Función de envió de datos

```
fprintf (swit1, "comando1\r\n");
```

Función de recibo de datos

```
dato_recibido_software = getc(swit1);
```

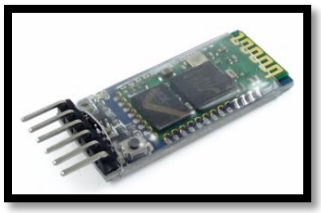
2. DESCRIPCION Y MANIPULACION DEL MODULO BLUETOOTH HC-05

2.1. DEFINICION

El modulo Bluetooth hc-05 es un módulo que permite comunicarnos inalámbricamente con un dispositivos que posea un conexión Bluetooth.

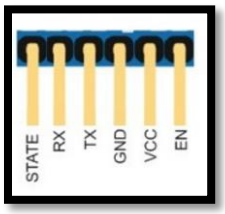
2.2. CARACTERISTICAS

- 3.3 / 5 v.Chip BC417143
- Alcance 10 mts
- Nivel TTL
- 1200bps a 1.3Mbps



2.3. PINES DE CONEXIÓN

- STATE, pin indicando cuando una conexión esta activa
- RX, pin de recepción de datos
- TX, pin de transmisión de datos
- EN, pin para cambiar el modo de operación se puede cambiar a modo Operación o modo Comandos AT



2.4. TABLA DE COMANDOS AT

	COMMAND	FUNCTION
1	AT	Test UART Connection
2	AT+RESET	Reset Device
3	AT+VERSION	Querry firmware version
4	AT+ORGL	Restore settings to Factory Defaults
5	AT+ADDR	Query Device Bluetooth Address
6	AT+NAME	Query/Set Device Name
7	AT+RNAME	Query Remote Bluetooth Device's Name
8	AT+ROLE	Query/Set Device Role
9	AT+CLASS	Query/Set Class of Device CoD
10	AT+IAC	Query/Set Inquire Access Code
11	AT+INQM	Query/Set Inquire Access Mode
12	AT+PSWD	Query/Set Pairing Passkey
13	AT+UART	Query/Set UART parameter
14	AT+CMODE	Query/Set Connection Mode
15	AT+BIND	Query/Set Binding Bluetooth Address
16	AT+POLAR	Query/Set LED Output Polarity
17	AT+PIO	Set/Reset a User I/O pin
18	AT+MPIO	Set/Reset multiple User I/O pin

19	AT+MPIO?	Query User I/O pin
20	AT+IPSCAN	Query/Set Scanning Parameters
21	AT+SNIFF	Query/Set SNIFF Energy Savings Parameters
22	AT+SENM	Query/Set Security & Encryption Modes
23	AT+RMSAD	Delete Authenticated Device from List
24	AT+FSAD	Find Device from Authenticated Device List
25	AT+ADCN	Query Total Number of Device from Authenticated Device List
26	AT+MRAD	Query Most Recently Used Authenticated Device
27	AT+STATE	Query Current Status of the Device
28	AT+INIT	Initialize SPP Profile
29	AT+INQ	Query Nearby Discoverable Devices
30	AT+INQC	Cancel Search for Discoverable Devices
31	AT+PAIR	Device Pairing
32	AT+LINK	Connect to a Remote Device
33	AT+DISC	Disconnect from a Remote Device
34	AT+ENSNIFF	Enter Energy Saving mode
35	AT+EXSNIFF	Exit Energy Saving mode

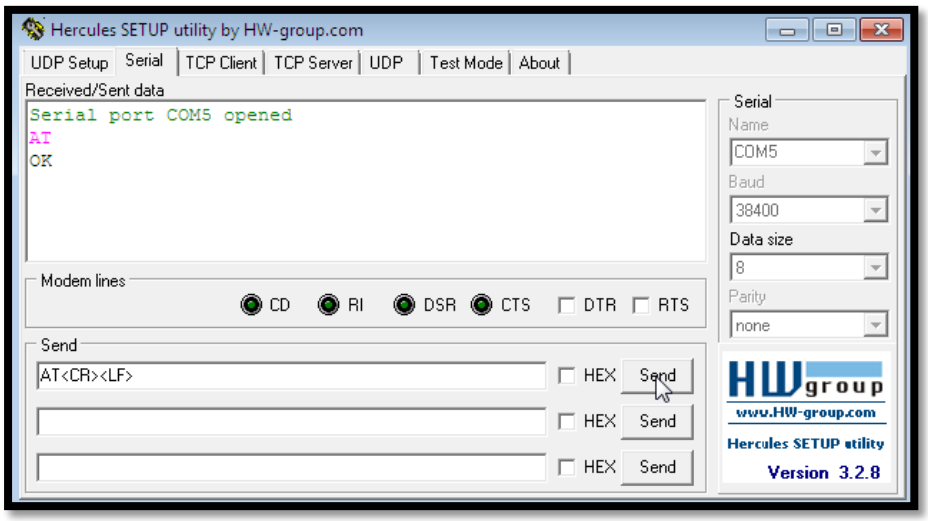
2.5. TABLA DE ERRORES

ERROR CODE	VERBOSE
0	Command Error/Invalid Command
1	Results in default value
2	PSKEY write error
3	Device name is too long (>32 characters)
4	No device name specified (0 lenght)
5	Bluetooth address NAP is too long
6	Bluetooth address UAP is too long
7	Bluetooth address LAP is too long
8	PIO map not specified (0 lenght)
9	Invalid PIO port Number entered
A	Device Class not specified (0 lenght)
B	Device Class too long
C	Inquire Access Code not Specified (0 lenght)
D	Inquire Access Code too long

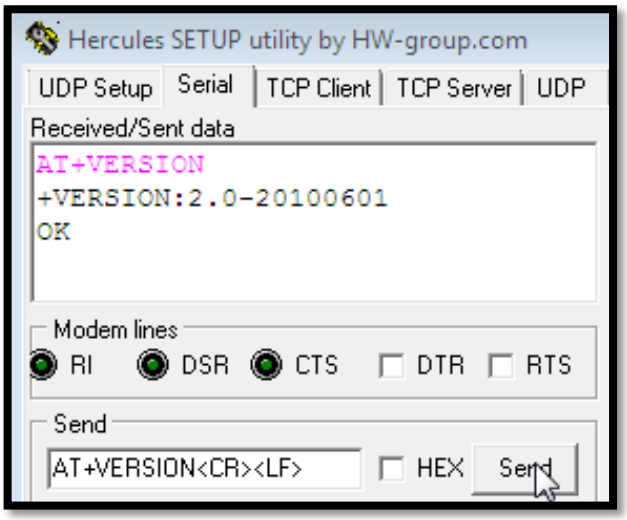
E	Invalid Iquire Access Code entered
F	Pairing Password not specified (0 lenght)
10	Pairing Password too long (> 16 characters)
11	Invalid Role entered
12	Invalid Baud Rate entered
13	Invalid Stop Bit entered
14	Invalid Parity Bit entered
15	No device in the Pairing List
16	SPP not initialized
17	SPP already initialized
18	Invalid Inquiry Mode
19	Inquiry Timeout occured
1A	Invalid/zero lenght address entered
1B	Invalid Security Mode entered
1C	Invalid Encryption Mode entered

2.6. CONFIGURACION PASO A PASO DEL MODULO BT HC-05

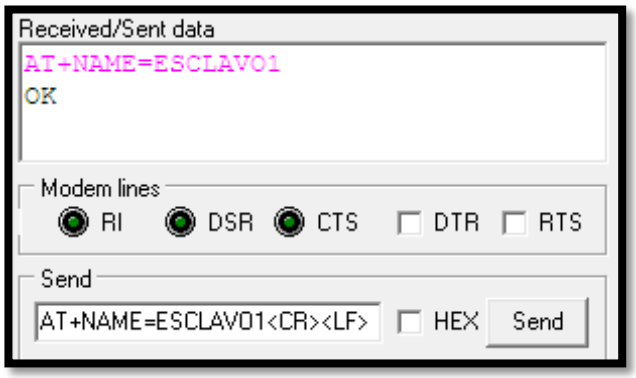
2.6.1.Testeo de la comunicación con el modulo Bluetooth HC-05



2.6.2.Comando para preguntar por la versión



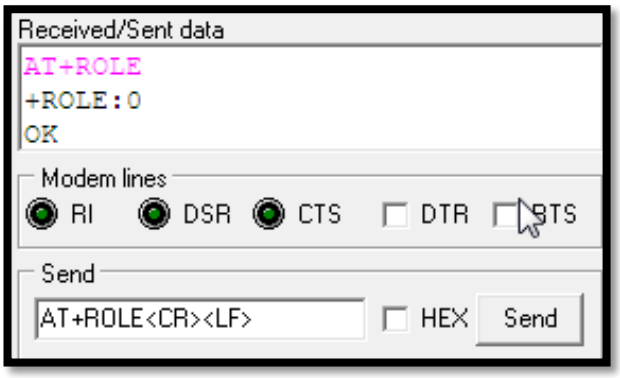
2.6.3.Comando para cambiar el nombre, **AT + NAME**



2.6.4.Comando para cambiar a modo maestro o esclavo, primero preguntamos cuál es su configuración: **AT+ROLE**

+ROLE = 0 >> SLAVE

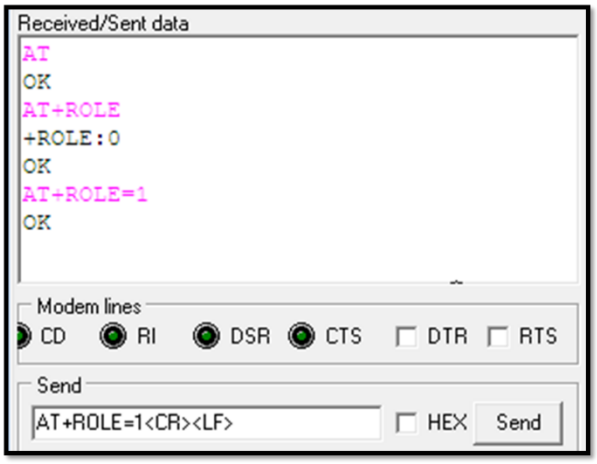
+ROLE = 1 >> MASTER



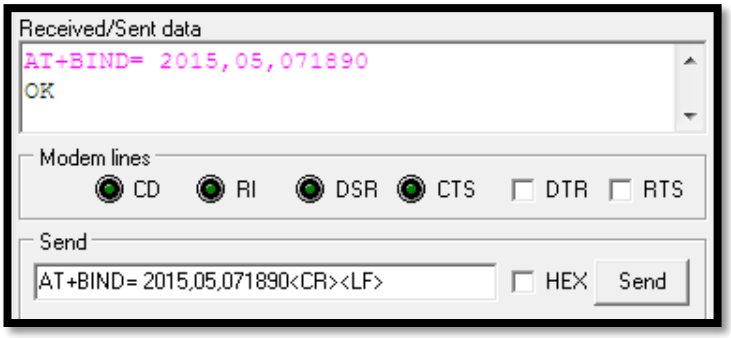
2.7. Configuración para enlazar el BT maestro con BT esclavo

2.7.1.OBTENCION DE LA MAC CON EL SOFTWARE BLUTERM

2.7.2.CONFIGURACION BT MODO MASTER



2.7.3.ENLAZE CON EL MODULO ESCLAVO

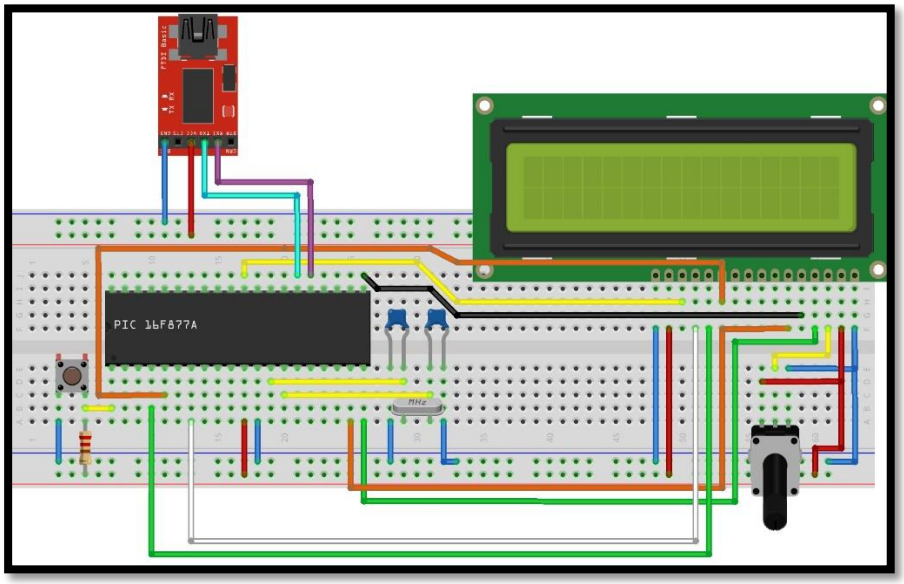


3. EJEMPLOS

3.1. Ejemplo1

Enviar la Lectura del canal análogo AN0 vía serial y recepcionar los datos enviados por la PC y mostrarlo en la LCD

3.1.1.Conexión



3.1.2.Programa

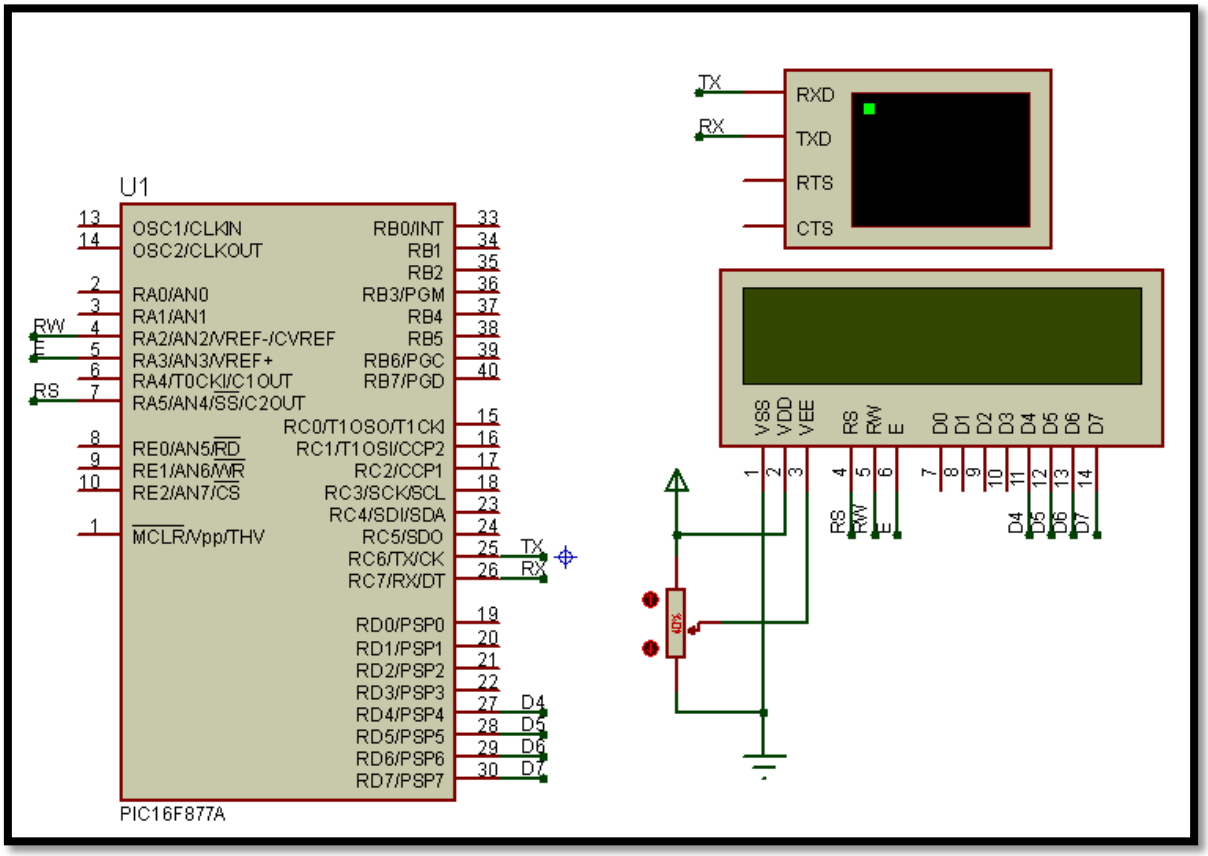
```
#INT_RDA
void recibir(){
   data = getc();
}

void main(){
   int16 adc1;

   setup_adc(  ADC_CLOCK_INTERNAL );
   setup_adc_ports(  AN0_AN1_AN3  );
   lcd_init();

   for(;;){
      set_adc_channel( 0 );
      delay_us(20);
      adc1 = read_adc();
      lcd_gotoxy(1,1);
      printf(lcd_putc,"Send Uart = %Lu",adc1);
      printf("Send Uart = %Lu\r\n",adc1);
      lcd_gotoxy(1,2);
      printf(lcd_putc,"Receive Uart: %c",data);
      delay_ms(100);
      lcd_putc('\f');
   }
}
```

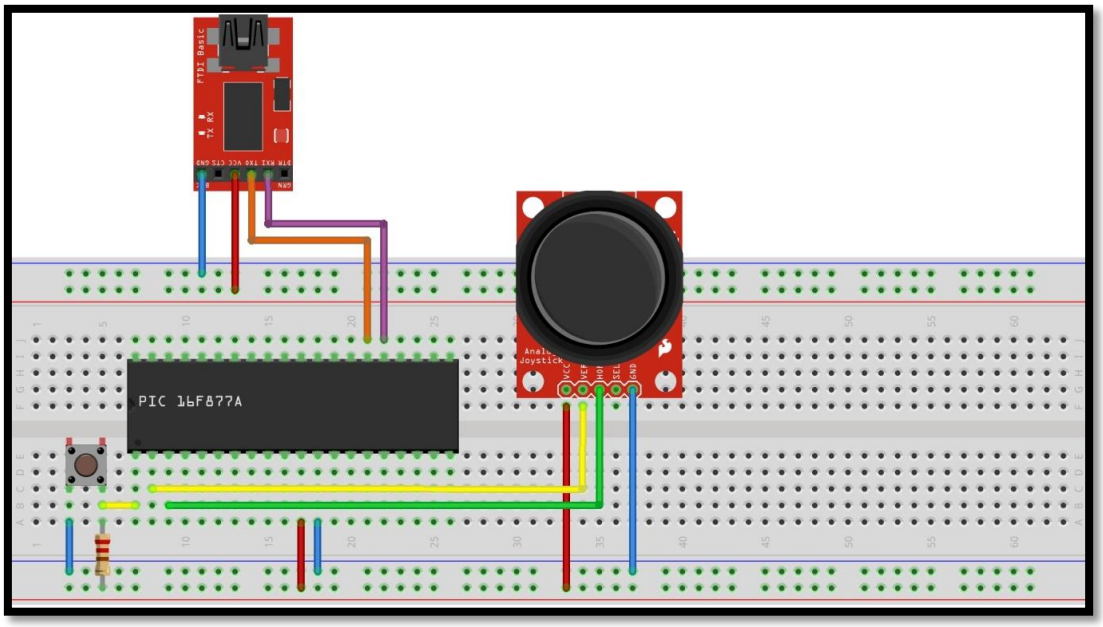
3.1.3.Simulación



3.2. Ejemplo2

Mediante el canal análogo AN0 y AN1 leer los valores enviados por un joystick , dichos valores deben ser enviados a la PC

3.2.1.Conexión



3.2.2.Código

Ejemplo2.c

```
#include <16f877a.h>
#device  adc=10
#fuses   XT
#use     delay(clock=4M)
#use     rs232(BAUD = 9600,XMIT = PIN_C6, RCV = PIN_C7)

void main(){
   int16 adc1,adc2;

   setup_adc(  ADC_CLOCK_INTERNAL );
   setup_adc_ports(  AN0_AN1_AN3  );

   for(;;){
      set_adc_channel( 0 );
      delay_us(20);
      adc1 = read_adc();

      set_adc_channel( 1 );
      delay_us(20);
      adc2 = read_adc();

      printf("x:%Lu - y:%Lu \r\n",adc1,adc2);

   }
}
```

3.3. Ejemplo3

Controlar un carro de dos llantas y una rueda loca mediantes ordenes enviadas por BT

3.3.1.Código

```
for(;;){
   if ( data == 'A'){   printf("Adelante\r\n");
      output_high(MOTOR1A); output_low(MOTOR1B);
      output_high(MOTOR2A); output_low(MOTOR2B);
      delay_ms(300);
   }
   if ( data == 'B'){   printf("ATRAS\r\n");
      output_low(MOTOR1A); output_high(MOTOR1B);
      output_low(MOTOR2A); output_high(MOTOR2B);
      delay_ms(300);
   }
   if ( data == 'c'){   printf("DERECHA\r\n");
      output_low(MOTOR1A); output_low(MOTOR1B);
      output_high(MOTOR2A); output_low(MOTOR2B);
      delay_ms(100);
      output_high(MOTOR1A); output_low(MOTOR1B);
      output_high(MOTOR2A); output_low(MOTOR2B);
   }
   if ( data == 'D'){   printf("IZQUIERDA\r\n");
      output_high(MOTOR1A); output_low(MOTOR1B);
      output_low(MOTOR2A); output_low(MOTOR2B);
      delay_ms(100);
      output_high(MOTOR1A); output_low(MOTOR1B);
      output_high(MOTOR2A); output_low(MOTOR2B);
   }
   if ( data == 'E'){  printf("APAGAR\r\n");
      output_low(MOTOR1A); output_low(MOTOR1B);
      output_low(MOTOR2A); output_low(MOTOR2B);
   }
}
```

3.3.2.Simulación