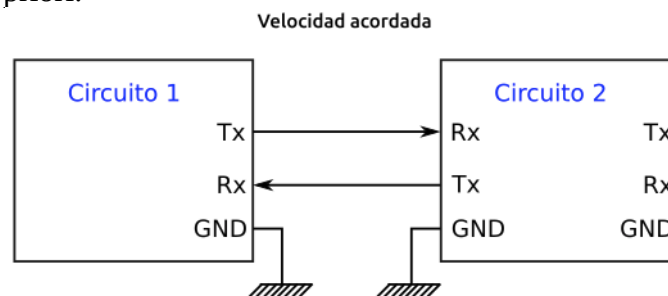


Mediante comunicaciones serie ASÍNCRONAS nos podemos comunicar también con dispositivos como ordenador, bluetooth-serie, Arduinos, ...

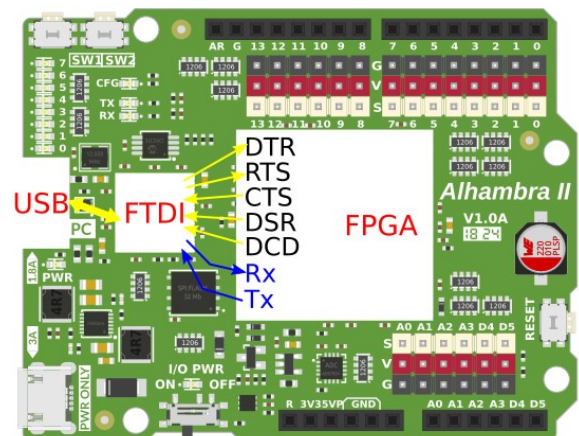
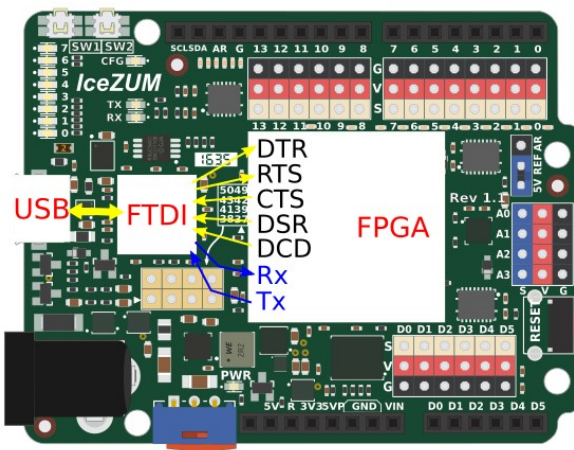
Cuando hablamos de puerto serie, usualmente usamos la Norma RS-232, que además de las comunicaciones serie asíncronas, define cosas como cables de control: DTR (*Data Terminal Ready*), RTS (*Request To Send*), CTS (*Clear To Send*), DSR (*Data Set Ready*), DCD (*Carrier Detect*) y RI (*Ring Indicator*).

En comunicaciones serie asíncronas utilizamos un único cable de datos transmisión (salida de información) que se llama TX, y otro de recepción (entrada de información) que se llama RX. No hay un cable con la señal de reloj, como en las comunicaciones serie síncronas por lo que la velocidad se acuerda a priori.



Los conectores DB9 clásicos para este tipo de comunicaciones no suelen usarse en los actuales ordenadores, pero dada su utilidad aparecen los chips FTDI que convierten las señales USB-serie. En concreto en las placas Alhambra es el FT2232H, que en realidad ofrece dos interfaces a través del USB:

1. Comunicaciones serie síncronas (bus spi) para realizar la carga de los circuitos
2. Puerto serie, realizando la conversión USB-serie y dando acceso a las señales de la norma RS-232 desde la FPGA.





En la tabla vemos un resumen de estos pines en Icestudio.

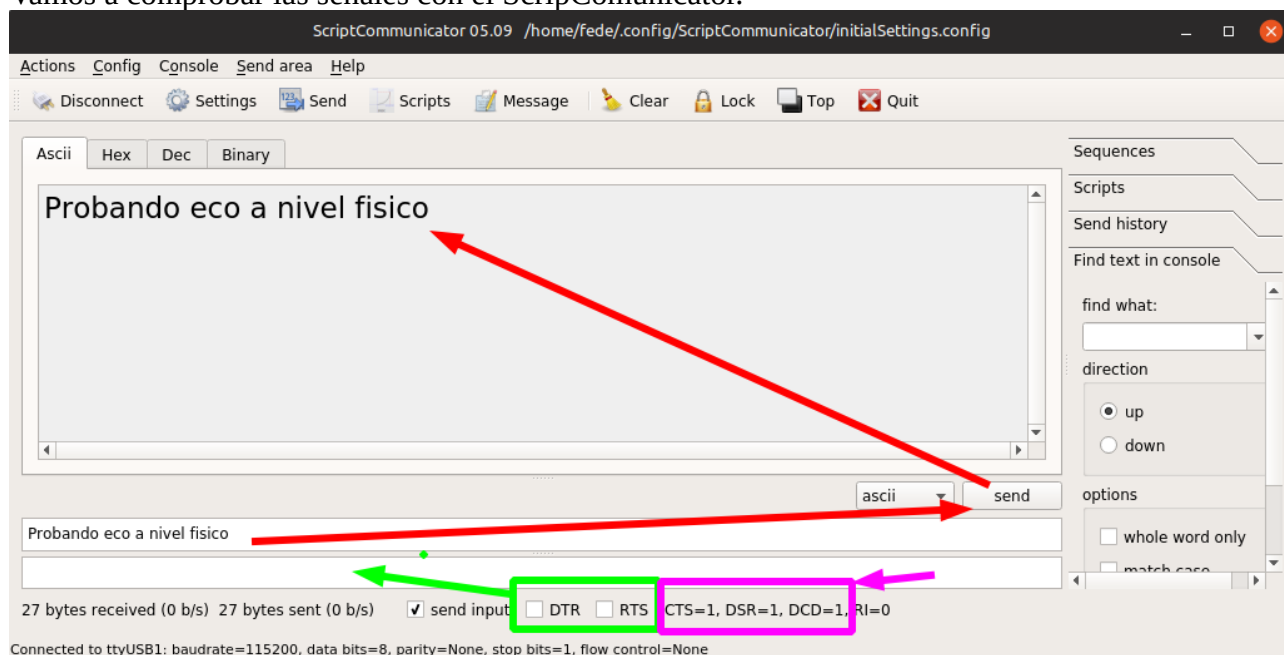
Pines del puerto serie		
	Entrada	Salida
Señales de control	RTS x ▼	CTS x ▼
	DTR x ▼	DSR x ▼
		DCD x ▼
Señales de datos	RX x ▼	TX x ▼

Terminales de comunicaciones

Usaremos el [Arduino-Ide](#) (1.8.7) y el [ScriptComunicator](#) (5.09), que son libres. Vamos a comprobar su funcionamiento para asegurar que todo está correcto. Mediante el ejemplo 5-03. Comprobaciones.

Ejemplo 5-03. Comprobaciones. Unir con cables y mostrar el estado de las señales de control para comprobar que están operativas.

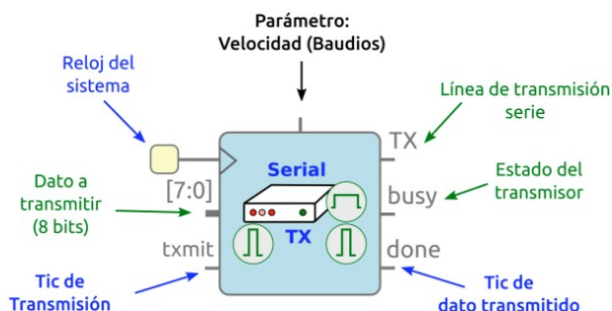
Vamos a comprobar las señales con el ScripComunicator.





Transmisor Serie

El envío de datos desde la FPGA al PC lo realizamos a través del transmisor serie. El bloque de Icestudio es este

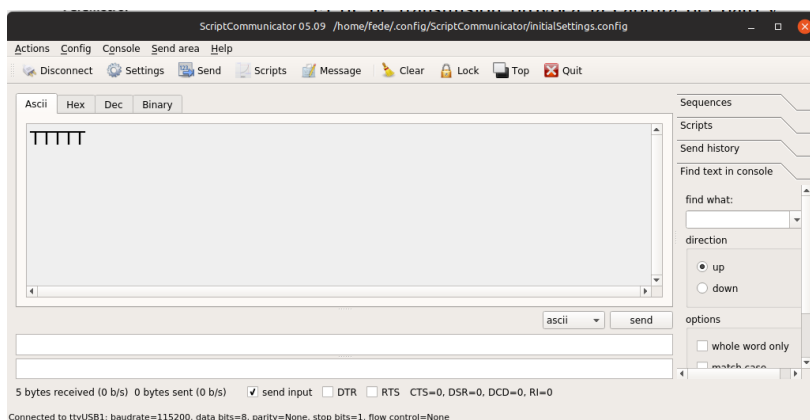


El tic de transmisión provoca la captura del dato y su envío por TX.

Busy indica el estado del transmisor e impide que un dato se transmita si no ha finalizado el anterior. Una vez enviado el dato se emite el tic de dato transmitido.

Velocidades de transmisión en baudios (parámetro): 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600 ó 115200. Por defecto 115200 baudios.

Ejemplo 5-04. Envío de un carácter. Cada pulsación de un pulsador va a provocar el envío del carácter T (0b0101 0100, 84 o 0x54) y Cuando se haya enviado, encenderemos 4 LEDs durante 300ms.



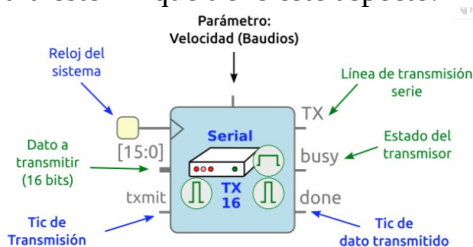
Ejemplo 5-05. Envío de un número en binario. Cada pulsación de un pulsador va a provocar el envío del número introducido mediante tres interruptores.

Ejemplo 5-06. Envío de un número decodificado de BCD a ASCII. El mismo ejemplo que el 5-05 pero decodificando el número de BCD a ASCII. Para ello los número por encima de 7 se activan pulsando SW1.

Ejemplo 5-07. Transmisor de 16 bits. Para transmitir un dato de 16 bits, hay que hacerlo en dos envíos de 8 bits. En este ejemplo enviamos dos caracteres, la "O" y "K". Primero enviamos el byte más significativo, y luego el menos. Cada vez que se apriete el botón se envían los dos caracteres.

Bloque transmisor de 16 bits

Se crea en Icestudio el bloque para este fin que tiene este aspecto.

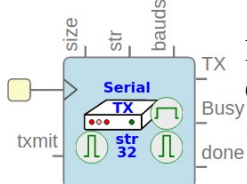


Ejemplo 5-08. Bloque transmisor de 16 bits. Mediante un switch se selecciona el dato a enviar. Cada dato son 2 caracteres. En una posición se envía el valor 0x3D3E, que se corresponde con => y en la otra posición se envía una A y una línea nueva (\n).

Envío de cadenas

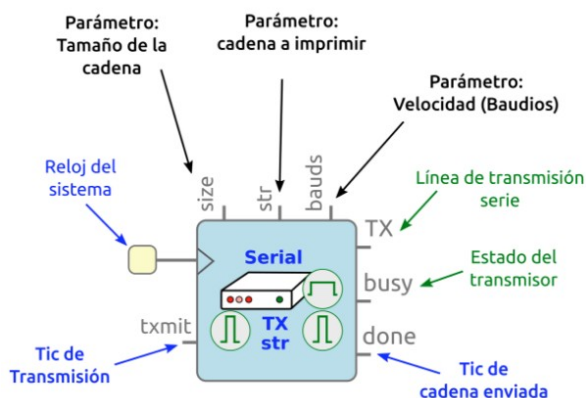
Un transmisor de cadenas se fundamenta en una tabla que contiene la cadena y un contador que se usa para recorrer la tabla.

El bloque para enviar una cadena máxima de 32 bytes (contador de 5 bits y una memoria de 32 bytes) es el siguiente.



Para enviar cadenas más largas sólo hay que crear el componente con un contador y memoria de mayor número de bits.

En Icestudio existe el componente serial-tx-str que es paramétrico (bloque está implementado en verilog). Nos permite enviar cadenas de cualquier longitud (siempre que no desbordemos el tamaño máximo de memoria interna).





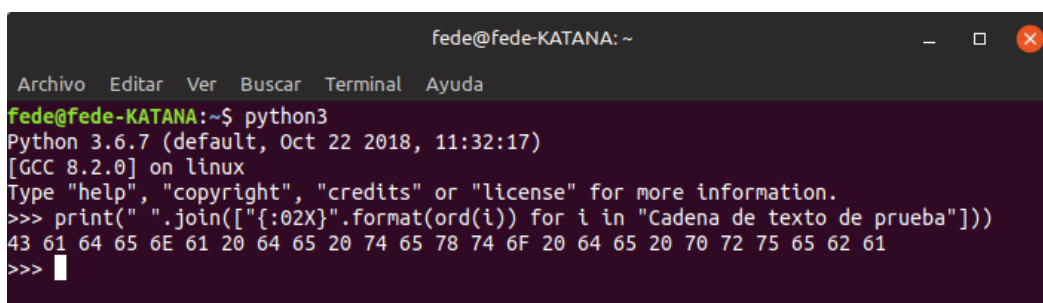
Ejemplo 5-09. Transmitir una cadena. Enviar una cadena introducida en una tabla cada vez que se accione el pulsador.

Aunque el ejemplo es perfectamente funcional es muy tedioso introducir una cadena en una tabla carácter a carácter. Por esto lo que haremos es crear la cadena en un fichero que debe estar en el mismo directorio en el que está el de Icestudio. El fichero se introduce como parámetro, y debe estar entre comillas dobles "" y su extensión debe ser .list. Es decir, trasladamos la información de una tabla a un fichero.

El siguiente paso es generar los datos de ese fichero mediante un programa externo como python que se encargue de generar los datos como cadenas de caracteres, pixeles, notas, ...

La ejecución de esta línea en python genera el resultado que vemos.

```
print(" ".join("{:02X}".format(ord(i)) for i in "Cadena de texto de prueba"))
```

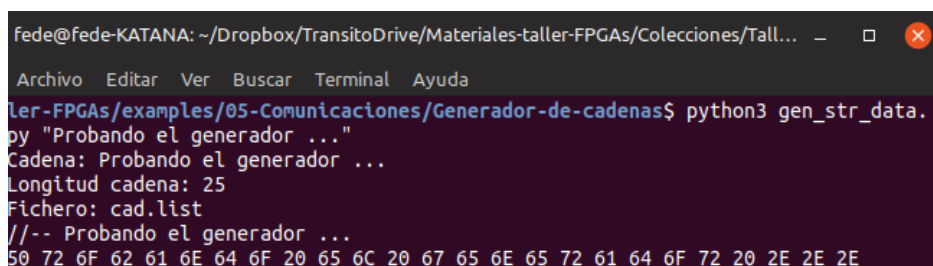


```
fed@fed-KATANA: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
fed@fed-KATANA:~$ python3  
Python 3.6.7 (default, Oct 22 2018, 11:32:17)  
[GCC 8.2.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> print(" ".join("{:02X}".format(ord(i)) for i in "Cadena de texto de prueba"))  
43 61 64 65 6E 61 20 64 65 20 74 65 78 74 6F 20 64 65 20 70 72 75 65 62 61  
>>> |
```

Los datos generados pueden copiarse en un fichero o en la memoria de Icestudio.

La mejor forma es hacer un programa completo como el realizado por Obijuan que genere un fichero a partir del parámetro que le pasemos.

El programa se puede descargar de aquí: [gen_str_data.py](#). Lo único que hay que hacer ahora es invocar al fichero desde la línea de comandos pasándole el dato como parámetro.



```
fed@fed-KATANA: ~/Dropbox/TransitoDrive/Materiales-taller-FPGAs/Colecciones/Tall...  
Archivo Editar Ver Buscar Terminal Ayuda  
ler-FPGAs/examples/05-Comunicaciones/Generador-de-cadenas$ python3 gen_str_data.  
py "Probando el generador ..."  
Cadena: Probando el generador ...  
Longitud cadena: 25  
Fichero: cad.list  
/-- Probando el generador ...  
50 72 6F 62 61 6E 64 6F 20 65 6C 20 67 65 6E 65 72 61 64 6F 72 20 2E 2E 2E
```

Todo listo para llevar el fichero generado al directorio adecuado y probar los resultados.

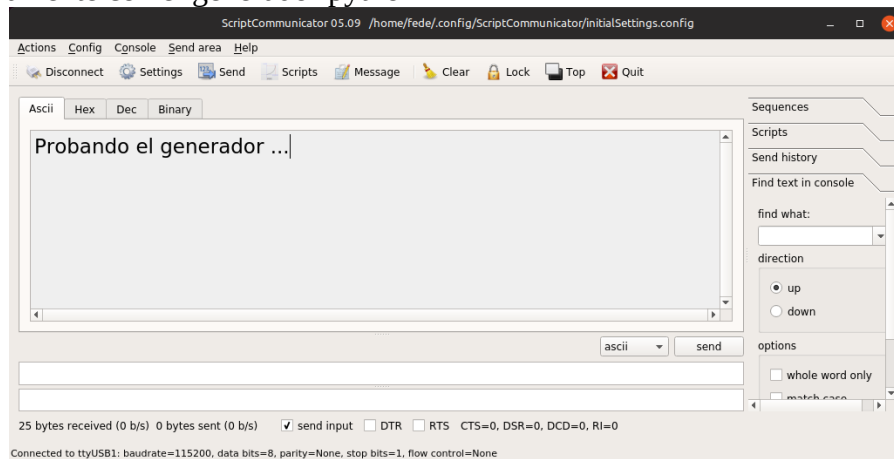


17

Comunicaciones: Puerto serie

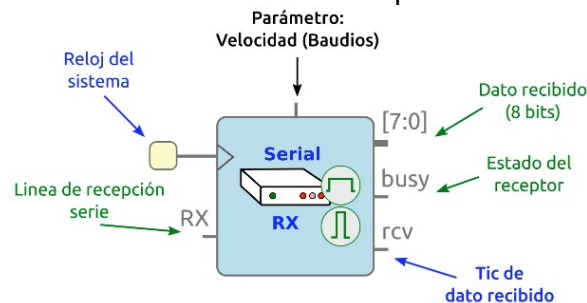


Ejemplo 5-10. Transmitir una cadena desde un archivo. Enviar una cadena desde un archivo generado previamente con el generador python

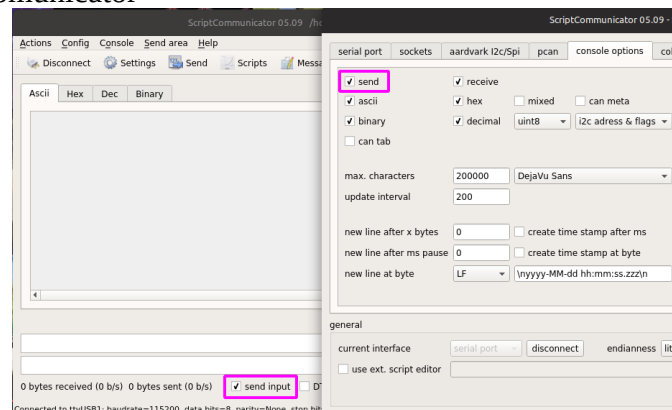


Receptor serie

Para recibir datos en la FPGA desde el PC usamos el receptor serie.



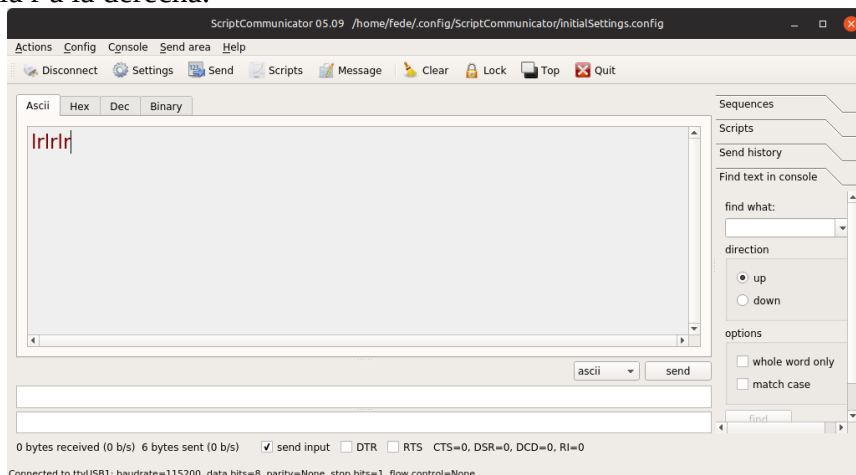
Ejemplo 5-11. Recepción de datos desde el PC. Leer todo lo que se envía desde el PC a la FPGA y mostrarlo en binario en los LEDs.
Configuramos ScriptCommunicator



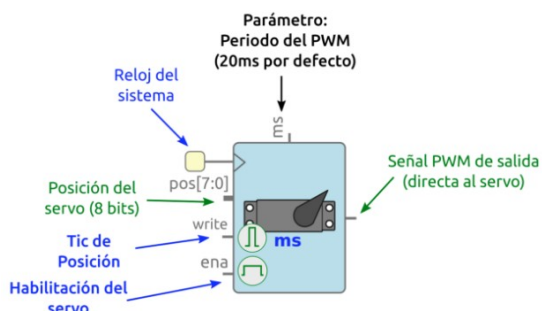


A través del puerto serie podemos usar nuestro teclado del PC para controlar circuitos.

Ejemplo 5-12. Mover un servo desde el teclado del PC. Con la tecla l moveremos el servo a la izquierda y con la r a la derecha.



Lo interesante sería poder mover el servo a cualquier posición, y esto se puede hacer con el bloque de Icestudio ServoPWM-8bits.



Para los servos Futaba 3003 las posiciones pueden estar entre los valores 60 y 225. El periodo de la señal PWM vale por defecto 20ms (el de la mayoría de los servos), pero se puede poner otro valor como parámetro, para que funcione con cualquier servo.

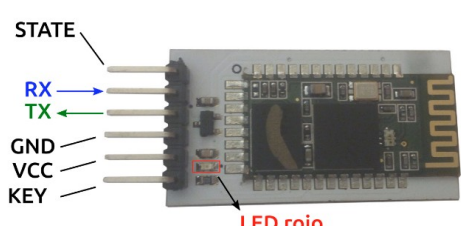
Ejemplo 5-13. Mover un servo a cualquier posición desde el teclado del PC. Enviando valores PWM entre 60 y 225 mover un servo futaba 3003.

Ejemplo 5-14: Eco. Todos los caracteres recibidos se vuelven a enviar.

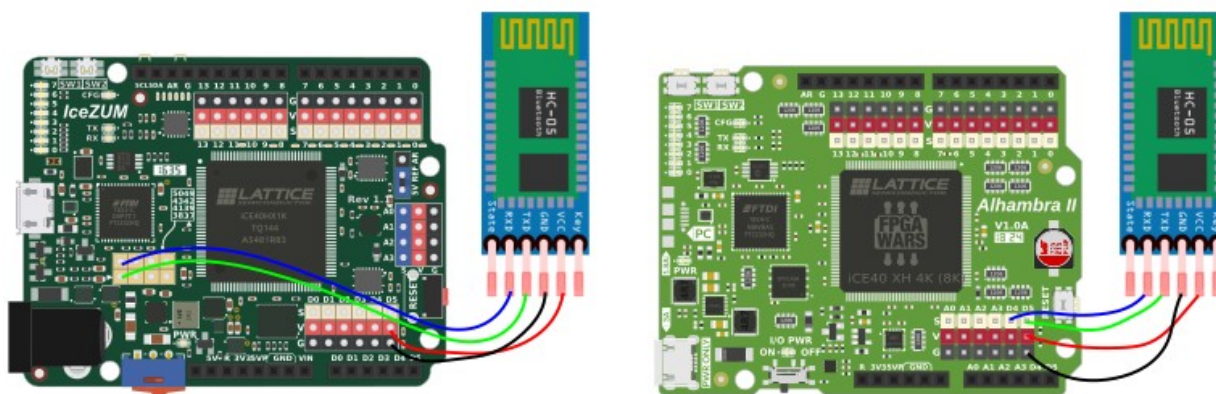
Ejemplo 5-15: Cifrador/Descifrador. Según el estado del interruptor externo, los datos recibidos por el puerto serie se cifran, o no, y se vuelve a enviar de vuelta al PC. El cifrado usado es muy básico: se intercambian los bits 0 y 1, y los 2 y 3.

Comunicación bluetooth-serie

Módulo HC-05:

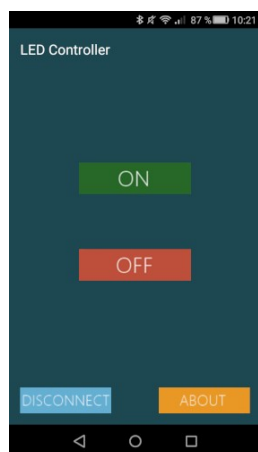
	State	Actividad transmisión/recepción. Pin opcional conectar un LED
	Key	Activar modo configuración del módulo
	VCC	Alimentación (3.3v - 5v)
	GND	Masa
	TX	Transmisión de datos (3.3v)
	RX	Recepción de datos (3.3v)

Conexionado en las placas.



Los módulos HC-05 vienen configurados con un firmware que normalmente trabaja a 38400 baudios. Para comenzar las pruebas vamos a usar nuestro móvil como terminal de comunicaciones, lo que resultará sumamente fácil con APPs como Bluetooth Terminal Qwerty o BlueTerm.

Ejemplo 5-16: Eco por bluetooth. Usaremos una APP terminal de bluetooth para comunicarnos con la FPGA. Usaremos el ejemplo eco adaptado.



Aunque para controlar servos y otros dispositivos lo más conveniente va a ser crearnos nuestra propia APP para nuestras pruebas utilizaremos una ya hecha: [Arduino Bluetooth Basic](#) de [Mayoogh Girish](#). Es una APP "hola mundo" que tiene sólo 2 botones, para encender y apagar un LED. Se trata de una APP libre: todo su código fuente está disponible en github, y tenemos la libertad para estudiar el código y modificarlo.



Ejemplo 5-17: Control de un servo combinado con bluetooth. Vamos a realizar el movimiento del servo Futaba 3003 a las posiciones izquierda, centro y derecha (Para servos futaba estas posiciones son 0x3C, 0xE1 y 0x80).

Se incluye una conexión de recepción a través de bluetooth que hará que los LEDs 0, 3 y 6 parpadeen a una frecuencia de 10Hz. Un LED amarillo permanecerá encendido mientras esto ocurre. La app del movil será LED y el efecto se produce al pulsar uno de los dos botones.