EVALUACIÓN Y ANÁLISIS DEL DESEMPEÑO A NIVEL FÍSICO DE UN SISTEMA DE COMUNICACIÓN DE DATOS DE CORTO ALCANCE VÍA RADIO EN 2.4 GHZ QUE UTILIZA CODIFICACIÓN MANCHESTER

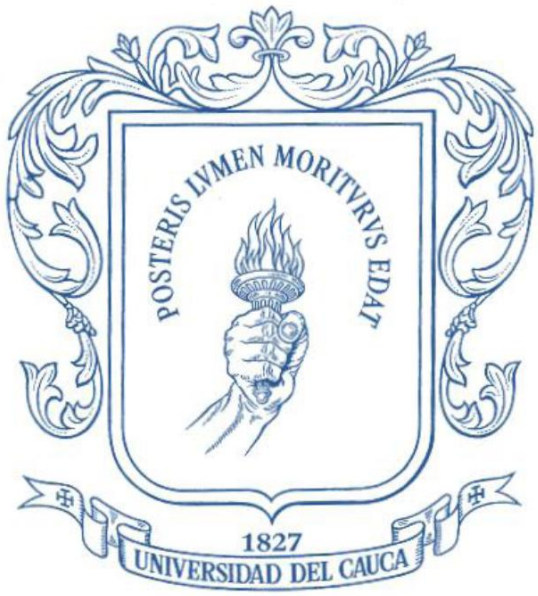
ANEXOS

MERY EVELYN CERON IBARRA
LUIS FERNANDO LEIVA BENACHI

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telecomunicaciones
GRIAL - Grupo de Radio e InALámbricas
GNTT – Grupo I+D Nuevas Tecnologías en Telecomunicaciones
Señales y Sistemas de Acceso y Difusión Basados en Radio
Gestión Integrada de Redes, Servicios y Arquitecturas de Telecomunicaciones
Popayán
2012

EVALUACIÓN Y ANÁLISIS DEL DESEMPEÑO A NIVEL FÍSICO DE UN SISTEMA DE COMUNICACIÓN DE DATOS DE CORTO ALCANCE VÍA RADIO EN 2.4 GHZ QUE UTILIZA CODIFICACIÓN MANCHESTER



ANEXOS

MERY EVELYN CERÓN IBARRA
LUIS FERNANDO LEIVA BENACHI

Trabajo de Grado presentado como requisito para obtener el título de Ingeniero en Electrónica y Telecomunicaciones

Director
Pablo Emilio Jojoa

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telecomunicaciones
GRIAL - Grupo de Radio e InALámbricas
GNTT – Grupo I+D Nuevas Tecnologías en Telecomunicaciones
Señales y Sistemas de Acceso y Difusión Basados en Radio
Gestión Integrada de Redes, Servicios y Arquitecturas de Telecomunicaciones
Popayán
2012

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

ANEXO A

MANEJO DE LAS HERRAMIENTAS SOFTWARE DEL KIT DE DESARROLLO CY8CKit-001 PSoC

La configuración de los dispositivos PSoC incluidos en el Kit de Desarrollo CY8CKit-001 de Cypress Semiconductor Corp se realiza mediante los programas PSoC Designer para el PSoC 1 y PSoC Creator para PSoC 3 y PSoC 5. En este anexo se describe el procedimiento para la creación y configuración de un proyecto, y la programación de los dispositivos PSoC.

En el Kit de Desarrollo CY8CKit-001 viene incluido el Disco Compacto (CD, *Compact Disc*) de instalación de las herramientas PSoC Designer y PSoC Creator. Las versiones en el momento que se adquiere el kit están desactualizadas. Para el desarrollo del trabajo de grado se utilizan las versiones PSoC Designer 5.1 y PSoC Creator 2.0[1], las cuales eran las más actuales en el momento del desarrollo del prototipo de comunicación.

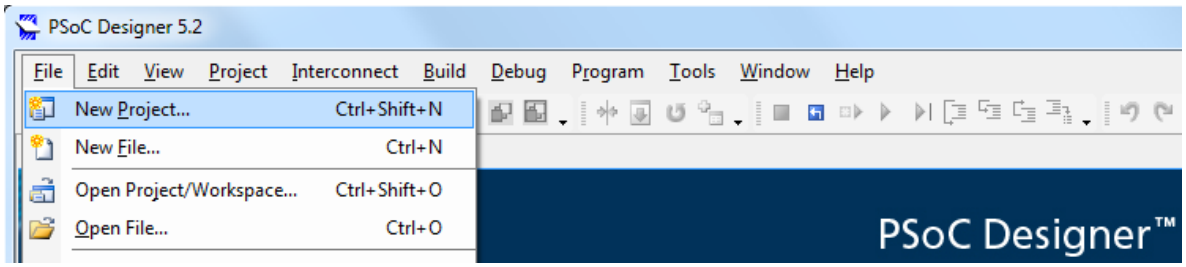
A.1. MANEJO DE LA HERRAMIENTA PSoC DESIGNER

Esta herramienta permite la creación de proyectos para el dispositivo CY8C28 (PSoC1) que se muestra en la Figura A.1.

Figura A.1. CY8C28

Para la creación de un proyecto, en el menú File se selecciona la opción New Project, como se muestra en la Figura A.2.

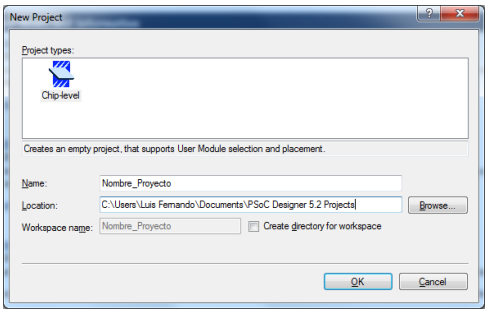
Figura A.2. PSoC Designer



[1] Las versiones recientes de los programas se pueden descargar de:
http://www.cypress.com/?rID=39531&source=home_documentation

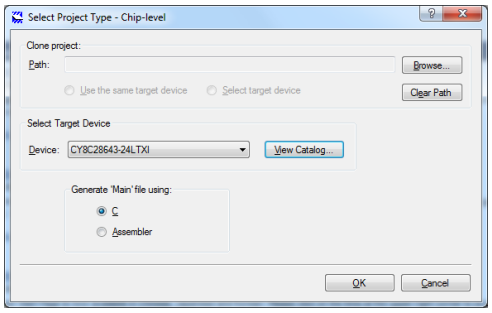
A continuación, se abre la ventana New Project donde se debe ingresar el nombre del proyecto y su localización, como se indica en la Figura A.3.

Figura A.3. Proyecto Nuevo PSoC Designer



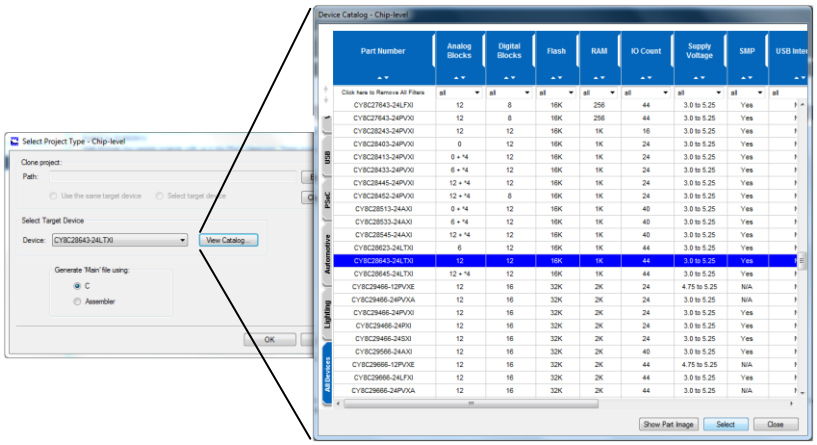
Después de ingresar la información se presiona OK y se abre la ventana Select Project Type. Aquí, se selecciona el dispositivo con el cual se va a desarrollar en proyecto y el leguaje de programación a utilizar (C o Assembler), como se muestra en la Figura A.4.

Figura A.4. Selección del Dispositivo



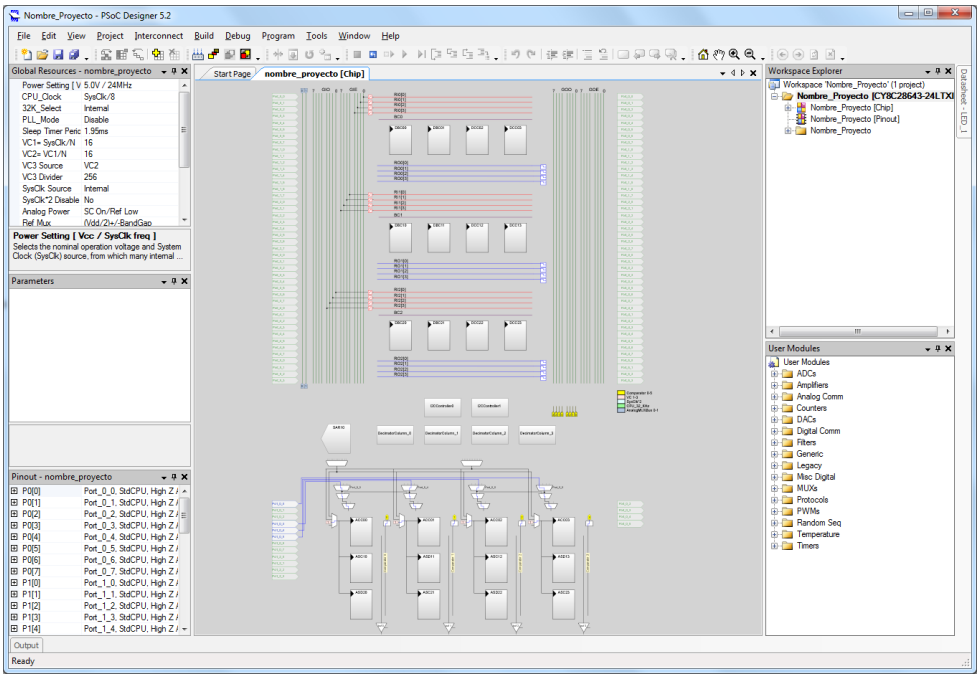
Para seleccionar el dispositivo base, existe un catalogo que se despliega al presionar View Catalog. Para el PSoC1 que se incluye en el kit se debe seleccionar la referencia CY8C28645-24LTXI como se muestra en la Figura A.5.

Figura A.5. Catalogo



Al terminar el procedimiento anterior, el proyecto se ha creado correctamente y se abre la ventana principal que contiene la vista del Chip Editor, en donde se diseña el proyecto a nivel del hardware interno en el PSoC.

Figura A.6. Chip Editor



Lo primero que se debe configurar son los recursos globales (*Global Resources*) del PSoC, como el voltaje, el reloj de la CPU, entre otros. La configuración se muestra en la Figura A.7.

Figura A.7. Recursos Globales

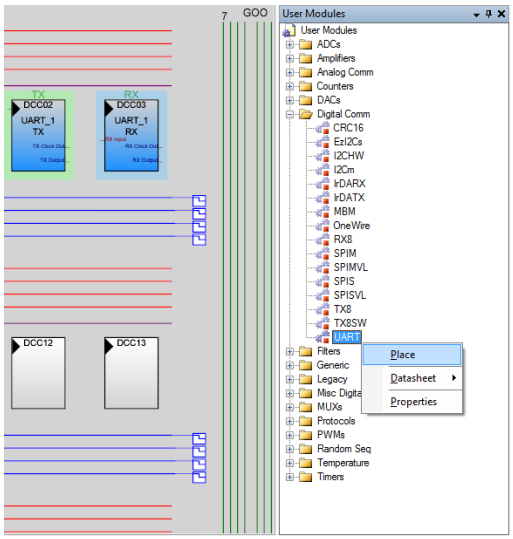
Global Resources - nombre_proyecto

Power Setting [Vcc / SysClk freq]	5.0V / 24MHz
CPU_Clock	SysClk/2
32K_Select	Internal
PLL_Mode	Disable
Sleep Timer Period	1.95ms
VC1= SysClk/N	16
VC2= VC1/N	16
VC3 Source	VC2
VC3 Divider	256
SysClk Source	Internal
SysClk*2 Disable	Yes
Analog Power	SC On/Ref Low
Ref Mux	(Vdd/2)+/-(Vdd/2)
AGndBypass	Disable
Op-Amp Bias	Low
SwitchModePump	OFF
Trip Voltage [LVD (SMP)]	4.81V (5.00V)
LVDThrottleBack	Disable
Watchdog Enable	Disable

Los módulos de usuario (*Usuario Modules*), son los recursos disponibles para el dispositivo CY8C28645-24LTXI. Para agregarlos al proyecto, se hace doble clic sobre el

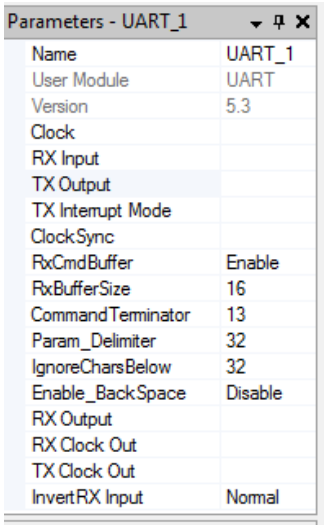
modulo o se presiona clic derecho y se selecciona Place como se muestra en la Figura A.8. Como ejemplo se adiciona el modulo UART.

Figura A.8. Módulos de Usuario



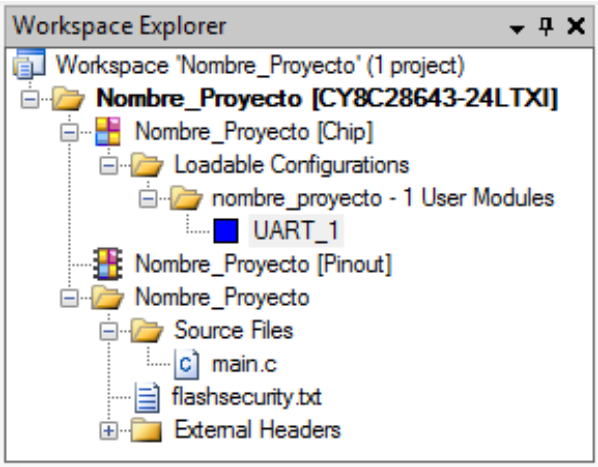
Adicionalmente, cada modulo cuenta con un Datasheet que contiene los parámetros configurables y las diferentes instrucciones de código que se pueden usar para la funcionalidad del mismo. Hay parámetros generales que se pueden configurar en la ventana principal como muestra la Figura A.9.

Figura A.9. Parámetros Generales



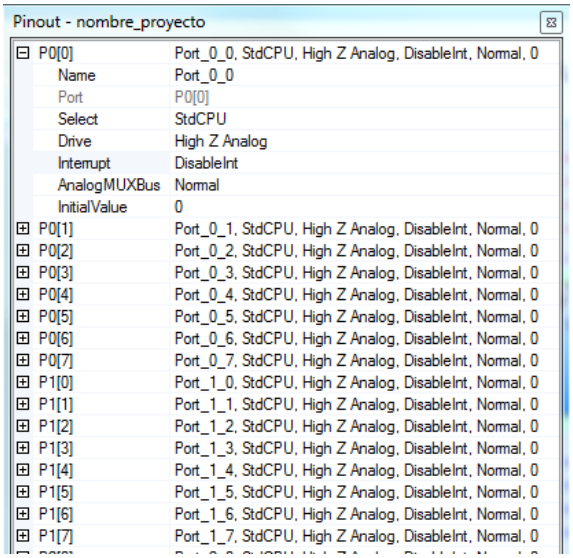
Los módulos de usuario, así como los demás elementos que pertenecen al proyecto aparecen en el Espacio de Trabajo (*Workspace*) en la ventana principal, como muestra la Figura A.10.

Figura A.10. Workspace Explorer



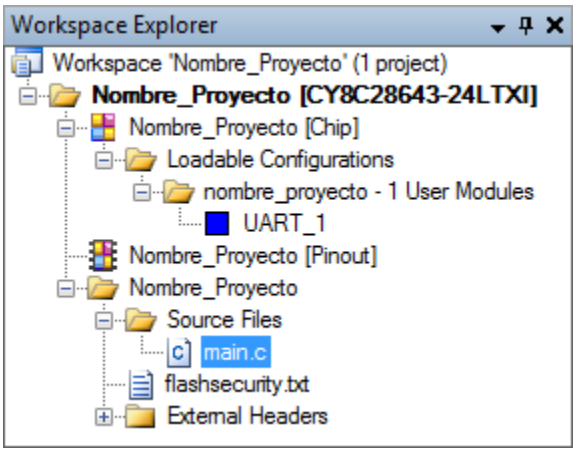
Los puertos del PSoC se configuran para que sean entradas, salidas o alta impedancia. También se puede configurar el tipo de interrupción de cada puerto, como se muestra en la Figura A.11.

Figura A.11. Pinout



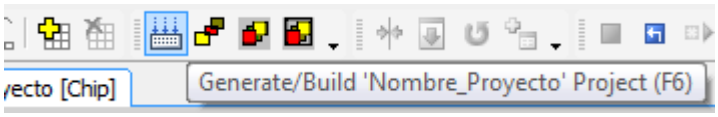
Una vez configurado todo lo correspondiente a hardware, se puede continuar con la creación de líneas de código donde se desarrollan las instrucciones que se ejecutaran en el proyecto. El archivo donde van las instrucciones es main.c y se encuentra en el Workspace, como muestra la figura A.12. Las funciones que puede ejecutar cada modulo de usuario se encuentran en su respectivo Datasheet.

Figura A.12. main.c



Para compilar el proyecto se presiona F6 o el botón *Generate/Build*, que se muestra en la Figura A.13. Al compilar el proyecto, se cargan las librerías usadas por los módulos de usuario del proyecto y se crean los archivos para la programación del PSoC.

Figura A.13. Compilación del Proyecto

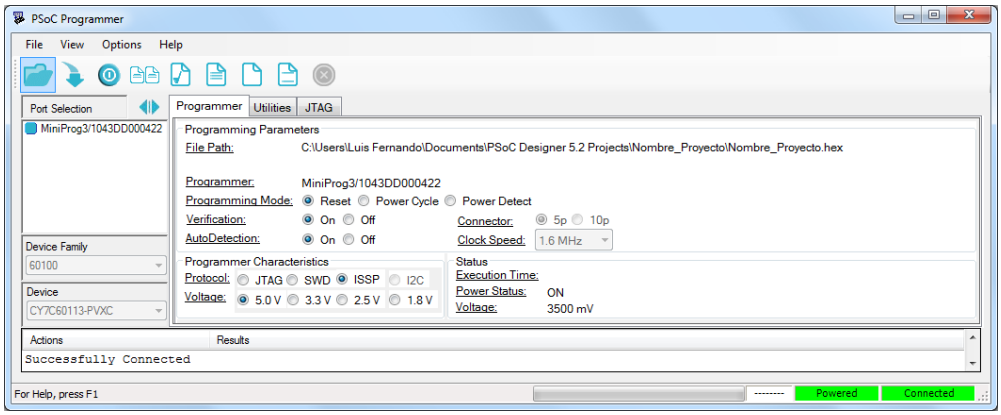


Para la programación del PSoC1 se utiliza el software PSoC Programmer, que tiene la función de cargar la compilación del proyecto (archivo de extensión .hex) al microcontrolador. Antes de programar, se debe conectar la tarjeta de desarrollo a la fuente de energía y el PSoC1 al computador a través del MiniProg3 como muestra la Figura A.14. De esta forma PSoC Programmer detectara automáticamente el dispositivo y se podrán configurar los parámetros de programación.

Figura A.14. Conexión MiniProg3

Después, se selecciona la ubicación del archivo de extensión *.hex* que se encuentra en la carpeta del proyecto presionando el botón File Load y se configuran los parámetros de programación como muestra la Figura A.15.

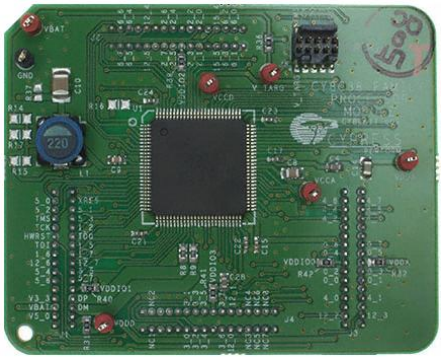
Figura A.15. PSoC Programmer



A.2. MANEJO DE LA HERRAMIENTA PSoC CREATOR

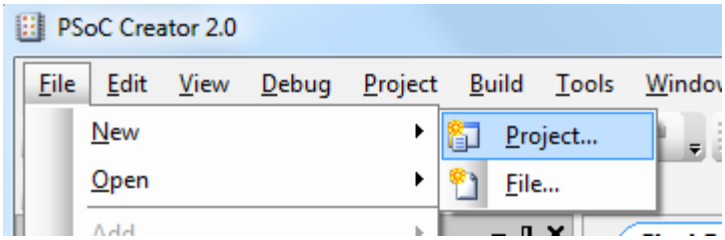
Esta herramienta permite la creación de proyectos para los dispositivos CY8C38 (PSoC3) y CY8C55 (PSoC5) que se muestran en la Figura A.16. El procedimiento para la creación de proyectos para los dos dispositivos es el mismo.

Figura A.16. CY8C38



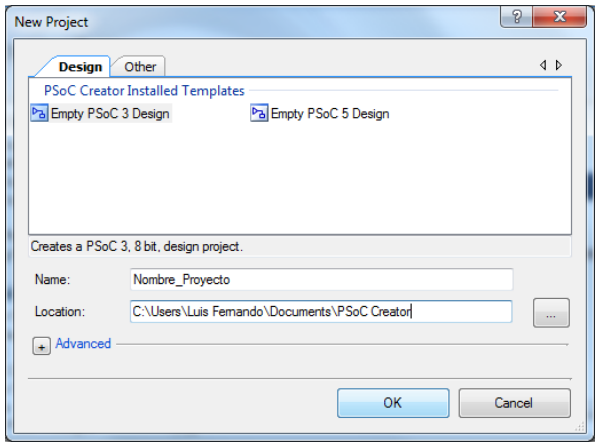
Para crear un proyecto, en el menú File se selecciona la opción New y Project, como se muestra en la Figura A.17.

Figura A.17. PSoC Creator



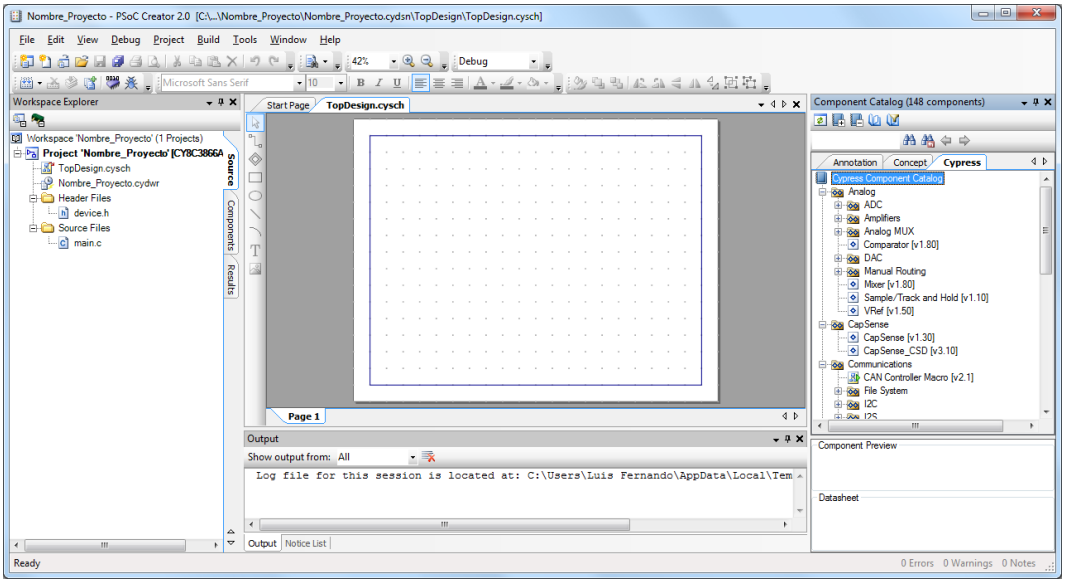
A continuación, se abre la ventana New Project donde se debe seleccionar entre PSoC3 y PSoC5 dependiendo para cual se creará el proyecto, después se ingresa el nombre y la ubicación del proyecto como muestra la Figura A.18.

Figura A.18. Proyecto Nuevo PSoC Creator



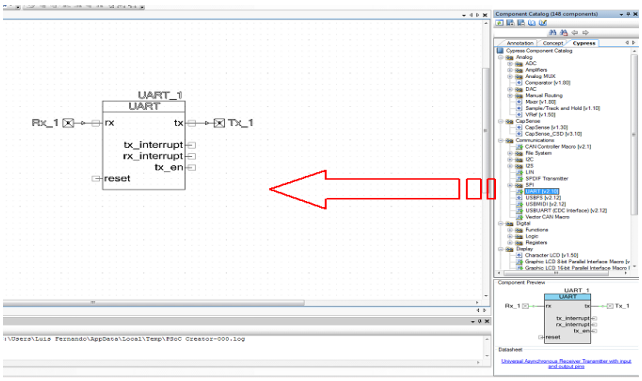
Al presionar OK, el proyecto se ha creado correctamente y se abre la ventana principal donde se diseña el proyecto.

Figura A.19. Ventana Principal PSoC Creator



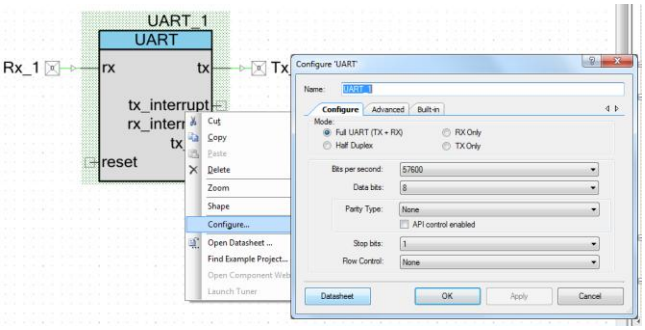
En el catalogo de componentes (Component Catalog), están los recursos disponibles para agregar al proyecto, para esto se seleccionan y se arrastran hasta el esquema del diseño como muestra la figura A.20. Como ejemplo se adiciona el modulo UART.

Figura A.20. Catalogo de Componentes



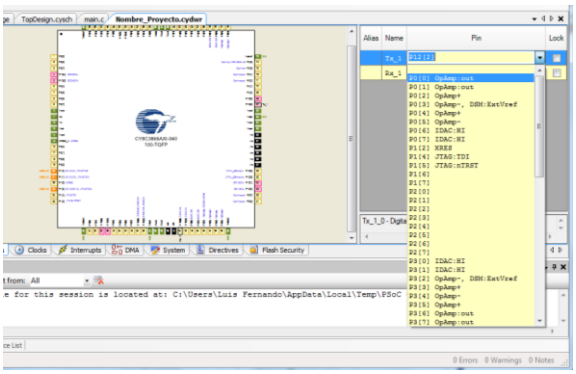
A cada modulo se le pueden configurar diferentes parámetros para modificar su funcionamiento según sea necesario. Para esto se presiona clic derecho sobre el modulo y se selecciona Configurar (*Configure*), como se observa en la Figura A.21. Para obtener más información sobre el modulo, se puede consultar el Datasheet disponible en la ventana de configuración.

Figura A.21. Configuración del Modulo



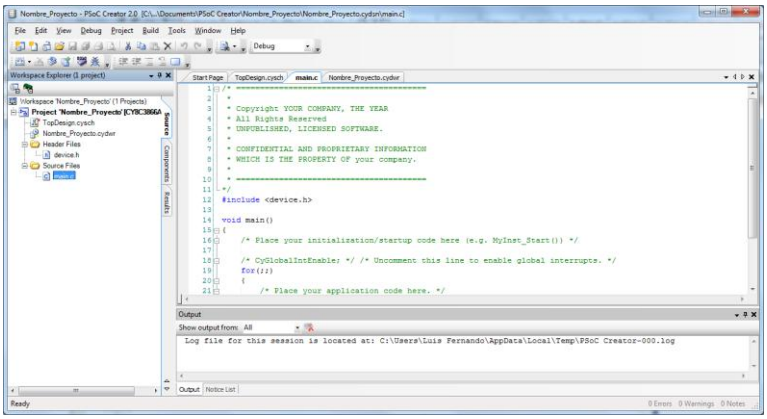
Para terminar el diseño del proyecto, se agregan componentes como pines de entrada y salida, señales de reloj, entre otros. Para asignar puertos de la tarjeta de desarrollo a las entradas y salidas del proyecto, en el Workspace se selecciona el archivo con extensión *.cydwr* para ingresar a la ventana de selección de pines, como se observa en la Figura A.22.

Figura A.22. Selección de Pines



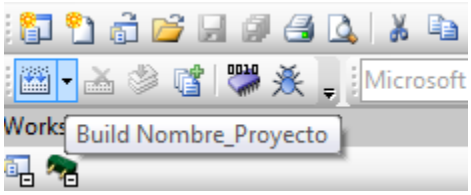
Una vez configurado todo lo correspondiente a hardware, se puede continuar con la creación de líneas de código donde se desarrollan las instrucciones que se ejecutaran en el proyecto. El archivo donde van las instrucciones es main.c y se encuentra en el Workspace, como muestra la Figura A.23.

Figura A.23. main.c



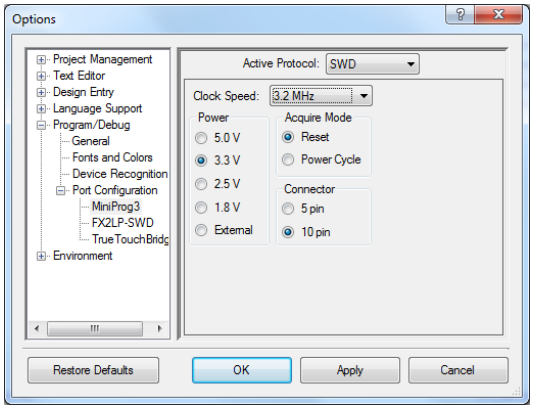
La compilación de proyecto se hace presionando Shift+F6 o el botón Build, que se muestra en la Figura A.24.

Figura A.24. Compilación del Proyecto



Para la programación del PSoC, primero se deben configurar los parámetros de programación del MiniProg3. Para esto, en el menú Tools y se selecciona Options. Aparece la ventana, en la cual se configuran los parámetros del MiniProg3 como muestra la Figura A.25 y se presiona OK.

Figura A.25. Parámetros MiniProg3

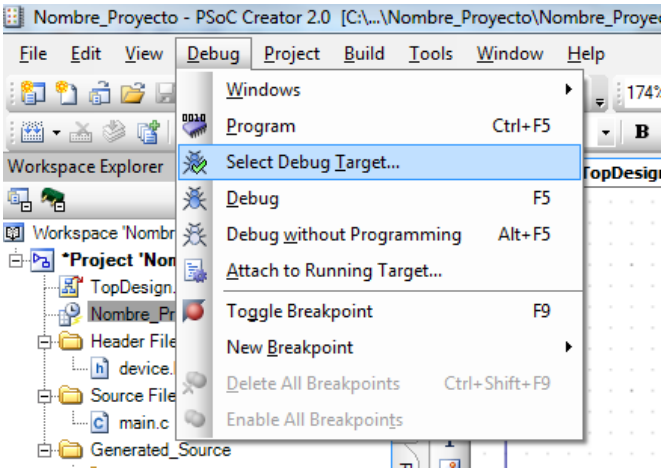


Ahora, se debe seleccionar el dispositivo por el cual la tarjeta de desarrollo está conectada al PC. Para esto, conectamos la tarjeta a la fuente de energía y el PSoC al PC como muestra la Figura A.26.

Figura A.26. Conexión MiniProg3

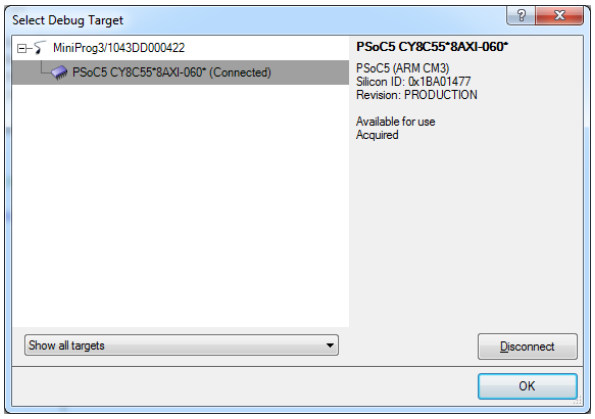
En el menú Debug se selecciona Select Debug Target, como se observa en la Figura A.27.

Figura A.27. Selección de Tarjeta



En la ventana se selecciona MiniProg3 y se presiona Port Acquire y después Connect, como se muestra en la Figura A.28.

Figura A.28. Conexión con Puerto



Finalmente, en el menú Debug se selecciona Program.

ANEXO B

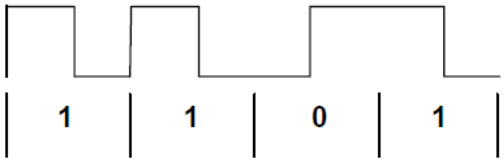
CODIFICACIÓN Y DECODIFICACIÓN MANCHESTER EN EL KIT DE DESARROLLO CY8CKIT-001

B.1. CODIFICADOR MANCHESTER

La codificación Manchester es un código de línea utilizado en sistemas digitales para brindarle robustez a la señal a transmitir. Esta codificación combina la señal de información y la señal de reloj en una sola, de esta forma, la señal se compone de transiciones que representan los datos de información así: Un 1 binario es representado por una transición alto-bajo a la mitad del periodo del bit. Y un 0 binario se representa por una transición bajo-alto a la mitad del periodo del bit.

En la figura B.1 se muestra un ejemplo de la codificación de una cadena de bits.

Figura B.1. Ejemplo codificación Manchester [1]



En esta sección del Anexo se muestra la forma como se implementa esta codificación utilizando los módulos programables disponibles en el PSoC Creator 2.0 [2] para una cadena de bytes.

IMPLEMENTACIÓN

La implementación parte del hecho, que la codificación Manchester es básicamente la combinación de la señal de información y la señal de reloj.

Se debe crear un nuevo proyecto en PSoC Creator como se explica en el anexo A. Para este caso se trabajo sobre el PSoC 5.

Para la implementación del codificador Manchester se utiliza básicamente el módulo de Interfaz Periférico Serial en modo Maestro SPIM (Serial Peripheral Interface) [3] y una compuerta XOR, conectados como se muestra en la Figura B.2

Los parámetros del módulo SPIM se configuran como se muestra en la Figura B.3.

Figura B.2. Codificador Manchester

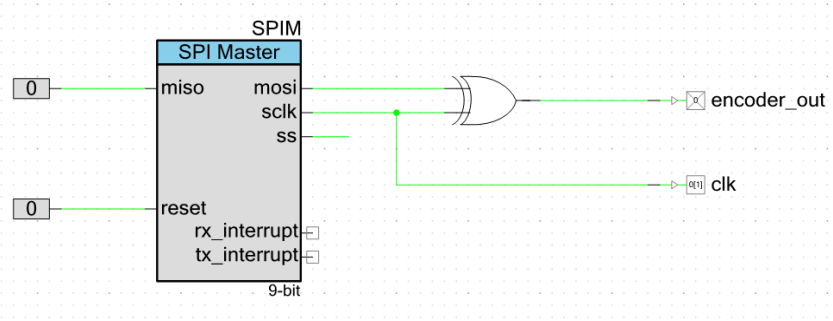
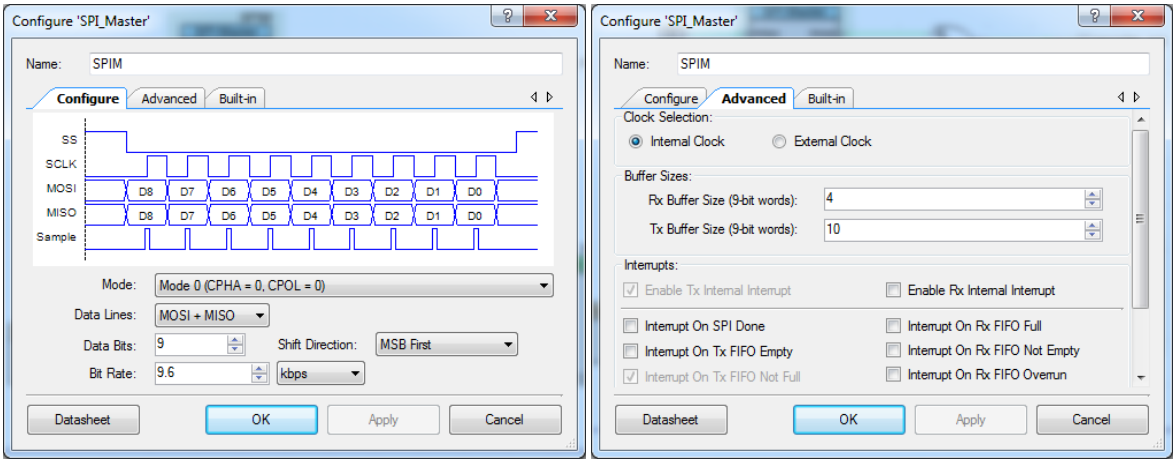


Figura B.3. Configuración de parámetros módulo SPIM



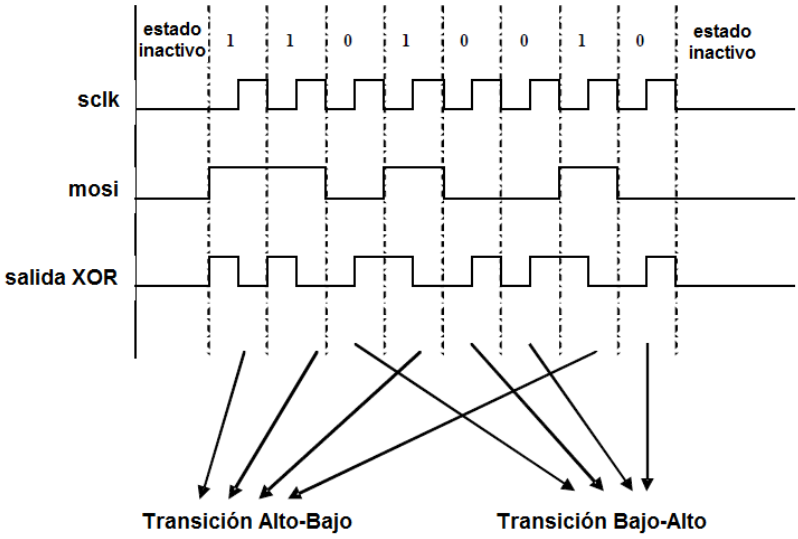
Cuando el SPI Master es configurado en “Mode 0”, el dato se establece en el flanco de caída de la señal de reloj y se fija en el flanco de subida de la señal de reloj. El estado inactivo de la señal de reloj es bajo. El pin MISO del SPIM se establece en bajo para que el estado de inactividad del pin MOSI sea bajo.

En las salidas mosi y sclk del SPIM, se aplica la operación XOR, de esta forma, a la salida de la compuerta XOR se obtiene los datos con codificación Manchester. Asi:

- Cuando la Salida MOSI es baja, la salida de la XOR es como la señal de reloj, es decir una transición bajo-alto (equivale a un dato “0”).
- Y cuando la salida MOSI es alta, la salida de la XOR es la señal inversa a la señal de reloj, por lo tanto hay una transición alto-bajo (equivale a un dato “1”).

EL diagrama de temporización del codificador se muestra en la Figura B.4.

Figura B.4. Diagrama de temporización del codificador Manchester



Se debe tener en cuenta que para una velocidad de bits determinada, configurada en el SPIM, se obtiene una señal codificada con el doble de velocidad. Para este caso, la tasa de bit es 9600 bps y la velocidad de los datos codificados es el doble, 19200 bps.

Interfaz con el módulo XBee-Pro

Para este trabajo de grado fue necesario desarrollar una etapa que sirviera de interfaz con el módulo XBee-Pro, que trabaja con el controlador UART[2]. Esto debido a que se presentó diferencias entre la señal entregada por el codificador y la señal esperada por el XBee-Pro, tal como el estado inicial de inactividad bajo del codificador y el bit de inicio y pare requerido por el protocolo serial del XBee.

La interfaz consta de un módulo “Shift Register” [4] y un módulo “UART” [5], dispuestos y conectados como se muestra en la Figura B.5. La configuración de estos módulos se hace como se muestra en la Figura B.6. Adicional a estos módulos de utilizó una compuerta NOT junto con un generador de señal de reloj (establecida en la velocidad de los datos codificados a capturar 19200 bps), que permitió habilitar el Shift Register solo cuando habían datos codificados a la salida. Para esto se hizo uso de la salida SS del SPIM.

El Shift Register, permitió tomar los datos a la salida del codificador y por medio del módulo UART enviarlos al XBee en un formato común.

[2] UART: Transmisor-Receptor Asíncrono Universal (Universal Asynchronous Receiver-Transmitter). Controlador utilizado en las comunicaciones seriales. Toma bytes de datos y transmite los bits uno a uno en forma secuencial. En la recepción nuevamente ensambla los bits en un byte de información.

También se configuró el LCD de la tarjeta para visualizar el dato codificado enviado y detectar posibles errores en la codificación.

Figura B.5. Diagrama final del codificador Manchester

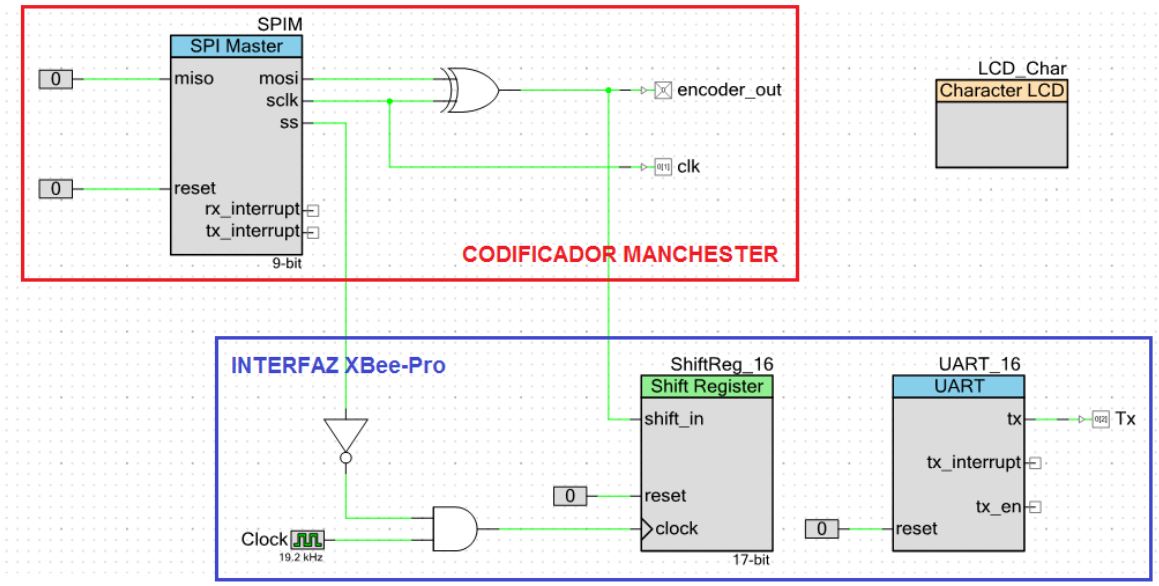
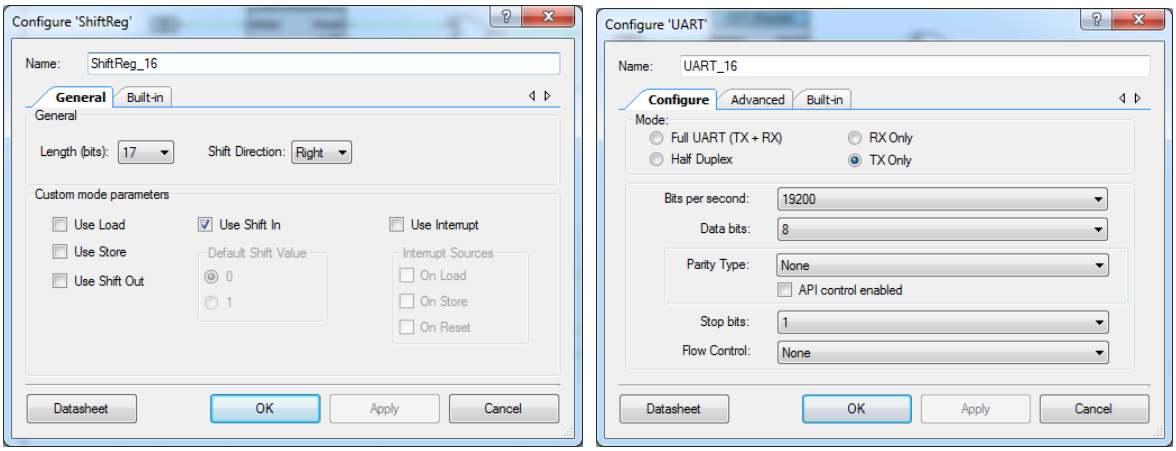
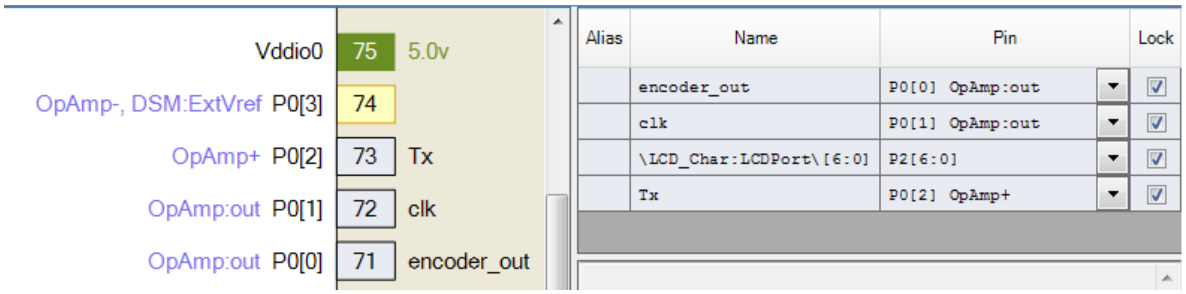


Figura B.6. Configuración Shift Register y UART



Seguidamente se debe hacer la configuración de pines para terminar con la etapa de la configuración hardware del proyecto. Para este trabajo la configuración de pines se hizo como se muestra en la Figura B.7. Se empleo el puerto 0 para las salidas, y el puerto 2 para el LCD.

Figura B.7. Configuración de pines para el codificador Manchester



Adicionalmente, se debe adicionar el código que controla el codificador Manchester en el archivo *main.c* que pertenece a los archivos fuente del proyecto. Las funciones empleadas para operar cada módulo se encuentran en el Datasheet de cada uno. En la Tabla B.1 se hace una breve descripción de las funciones utilizadas en este trabajo.

Tabla B.1. Funciones utilizadas en el Codificador Manchester

Módulo	Función	Descripción
SPIM	SPIM_Start()	Se debe llamar en el inicio para inicializar el componente.
SPIM	SPIM_WriteByte()	Coloca un byte en el buffer de transmisión que se enviará en el siguiente intervalo de tiempo disponible.
ShiftReg	ShiftReg_Start()	Inicial el componente y habilita todas las interrupciones seleccionadas
ShiftReg	ShiftReg_ReadRegValue()	Lee el valor actual del registro de desplazamiento
UART	UART_Start()	Inicializa y habilita la operación del UART
UART	UART_PutArray()	Trae los datos desde una cadena de memoria y los ubica dentro del buffer de memoria para transmitirlos

Se implementó una aplicación que envía 5000 bytes con la letra “a”. Esta rutina, realiza la codificación byte por byte, toma los datos codificados en el *Shift Register,* muestra el dato codificado en el LCD, debido a que un byte codificado está conformado por 16 bits es necesario dividirlo en 2 bytes y luego enviar cada byte al XBee por medio del módulo UART. La rutina se muestra a continuación.

```
/* ========================================================================================
 * CODIFICADOR MANCHESTER
 * ========================================================================================*/
#include <device.h>
#include <string.h>
#include <stdio.h>

void main()
{
```

```c
char* array = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa";
uint16 codigo;                                    //Declaración de variables
uint8 cod_tx[2];
uint8 i=0;
uint8 j;
LCD_Char_Start();                                 //inicializa módulos utilizados
ShiftReg_16_Start();
SPIM_Start();
UART_16_Start();
CYGlobalIntEnable;                                //habilita interrupciones globales

while(i<100)
       {
       for(j=0;j<50;j++)                          //Permite realizar la codificación de
                                                  5000 bytes
              {
              SPIM_WriteByte(array[j]);           //Codifica 1 byte
              CyDelay(3);                         //Retardo milisegundos*2
              codigo=ShiftReg_16_ReadRegValue();  //Toma el dato codificado
              if(codigo!=0)
                     {
                     LCD_Char_PrintHexUint16(codigo);//Muestra en LCD dato el codificado
                     cod_tx[1]=codigo;           //Toma 8 bits LSB del dato codificado
                     codigo = (((codigo & 0x00FF) << 8) | ((codigo & 0xFF00) >> 8));
                     cod_tx[0]=codigo;           //Toma 8 bits MSB del dato codificado
                     UART_16_PutArray(cod_tx, 2); //Envía por el UART los dos Bytes que
                                                  conforman el dato codificado al XBee
                     }
              CyDelay(1);
              }
              i++;
       }
}
/*==========================================================================================*/
```

Finalmente, en la Tabla B.2 se detalla la correspondencia de pines para conectar el XBee-Pro con la tarjeta. Adicionalmente se conecta Vcc (Fuente de voltaje) y GND (tierra).

Tabla B.2. Conexión de la Tarjeta de desarrollo con XBee-Pro

Pin en tarjeta de desarrollo	Pin en el Xbee-Pro
Tx=P0[2]	DIN=4

B.2. DECODIFICADOR MANCHESTER

Los códigos Manchester son bastante utilizados en los sistemas de comunicación debido a la simplicidad, pues no requiere protocolo de alto nivel para su implementación, además de los beneficios adicionales como la auto-sincronización y la independencia del medio de transmisión.

Sin embargo, la decodificación Manchester es un proceso un poco más complejo, debido a que se requiere extraer la señal de reloj y la señal de información de una única señal recibida, basado en la reconstrucción digital e implementada en bloques digitales en PSoC Creator.

En la Figura B.8 se muestra la codificación Manchester de cada valor de bit. Se debe tener en cuenta que adicional a las transiciones representativas de cada bit, cuando se

transmite una cadena de bits, pueden ocurrir transiciones adicionales entre bits, como se muestra en la Figura B.9

Figura B.8. Codificación Manchester del bit 0 y 1

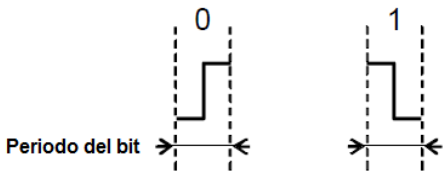
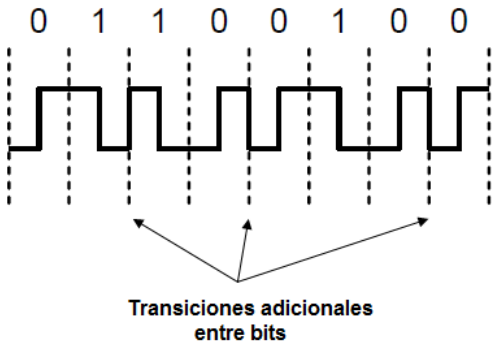


Figura B.9. Transiciones de una señal con codificación Manchester

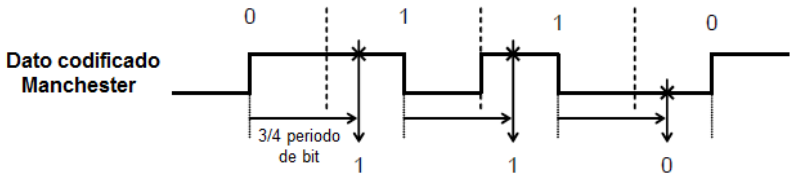


IMPLEMENTACIÓN

El método utilizado para realizar la implementación del decodificador Manchester, no se basa en la dirección de las transiciones que ocurren a mitad del periodo del bit, si no, en que el dato está contenido en la primera mitad del periodo del bit, antes de la transición [6].

De esta forma, un retardo de 3/4 del periodo del bit a partir del momento en que ocurre la transición a mitad del bit, indica en momento en que se capturar el valor del siguiente bit de información. Figura B.10.

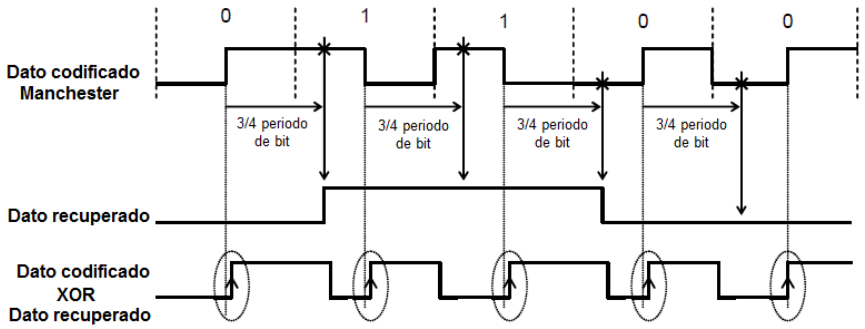
Figura B.10. Captura del valor del siguiente bit



El receptor establece una señal con estado inicial bajo, y si el siguiente bit es un 1, el receptor invierte la señal y produce una transición bajo-alto o mantiene la inversión de la señal si el anterior bit también era 1. Si el siguiente bit es un cero restablece la señal a su valor inicial bajo. Así, se obtiene una señal con el dato recuperado.

La señal de reloj es recuperada de la señal entrante de datos codificada, a partir de aplicar la operación XOR de la señal de datos codificada con la señal de datos recuperada. Figura B.11.

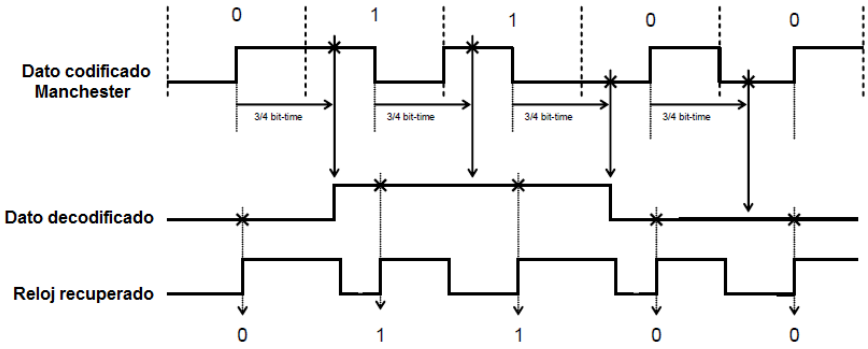
Figura B.11. Recuperación del dato y señal de reloj



Solo es necesario renombrar las señales, el Dato recuperado representa el dato decodificado o salida del decodificador Manchester, y la operación XOR representa la señal de reloj recuperada o de salida.

Se puede demostrar que el dato decodificado puede ser leído cada vez que ocurre un flanco de subida en la señal de reloj recuperada. Figura B.12.

Figura B.12. Decodificador Manchester



En un nuevo proyecto de PSoC Creator, se implementa en PSoC 5 el decodificador basado en la metodología anteriormente explicada. Para esto se requiere una compuerta XOR para generar la señal de reloj, un PWM para generar el retardo 3/4 del periodo del bit y un Flip-Flop tipo D[3] para registrar el dato de entrada cuando se cumple los 3/4 del periodo del bit que equivale al dato decodificado.

El esquema de la implementación del decodificador Manchester se muestra en la Figura B.12 y los parámetros del PWM se configuran como en la Figura B.13.

[3] Flip-Flop tipo D es un elemento de memoria que puede almacenar información en forma de un "1" o "0" lógicos. Este flip-flop tiene una entrada D y dos salidas Q y Q. Si hay un 1 o un 0 en la entrada D, la salida Q toma el valor y lo almacena con cada pulso de reloj.

Figura B.13. Decodificador Manchester

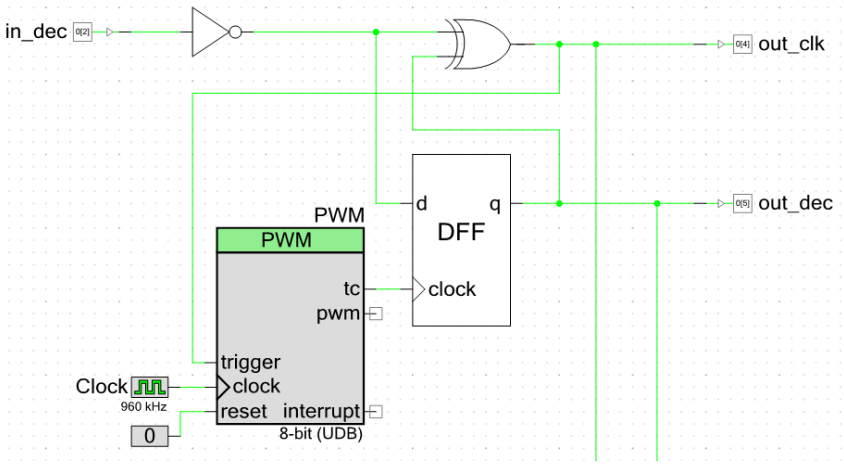
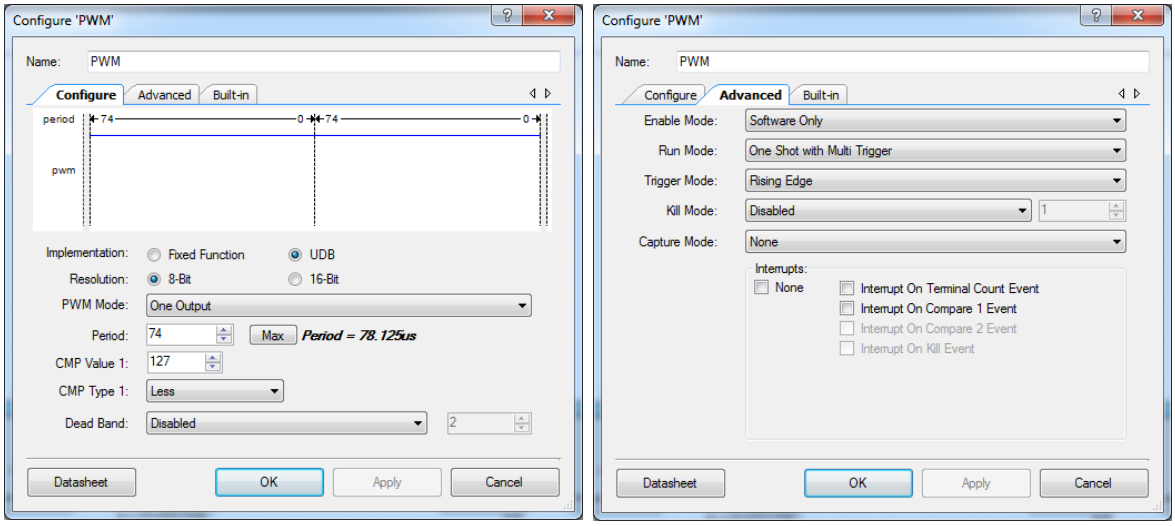


Figura B.14. Configuración de parámetros módulo PWM



Cuando ha pasado los 3/4 periodos, medidos por el PWM, se genera un impulso que va a la entrada de reloj del Flip-Flop D, en este momento la salida *q* se actualiza con el dato a la entrada *d*.

Para este trabajo se tiene que, el periodo de bit de los datos de información sin codificar corresponde a 1/9600, teniendo el periodo de bit, se determina que las 3/4 partes equivale a 78,125 µs, valor con el que se configura el módulo PWM.

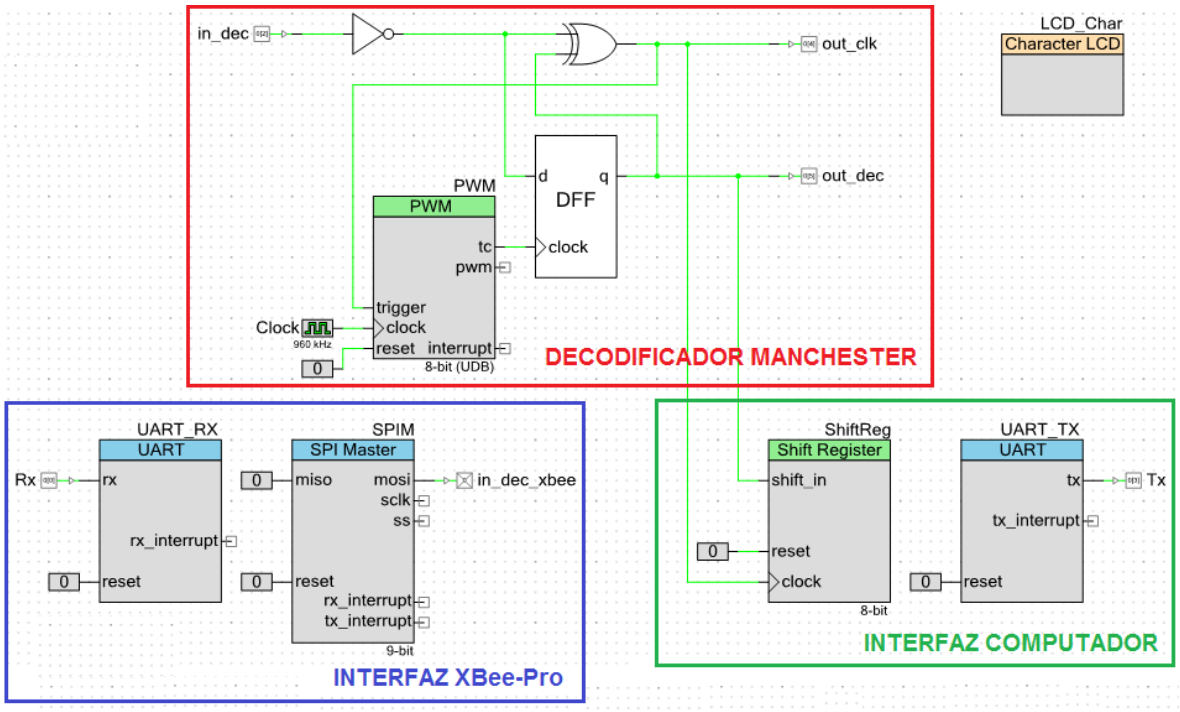
Cabe notar que la señal codificada entra al codificador a 19200 bps, y sale el dato decodificado a una velocidad de 9600 bps, velocidad inicial con la cual entra el dato al codificador.

Los datos decodificados se envían a un computador portátil por medio del puerto RS232 de la tarjeta, para evaluar la información recibida y posteriormente realizar análisis del sistema. Se hace uso de un cable TRENDnet TU-S9 [7] que convierte la señal proveniente de la tarjeta del puerto RS232, para ser apta para ingresar al computador portátil por medio del puerto USB.

Al igual que en el codificador, para el decodificador es necesario implementar etapas que sirvan de interfaz entre la tarjeta de desarrollo con el módulo XBee-Pro y con el computador.

El decodificador tiene algunas condiciones para realizar una correcta auto-sincronización, Tal como es el envió de un preámbulo donde se alterne unos y ceros. Pero con la implementación de la interfaz entre el XBee_Pro y la tarjeta, este problema no afecta el sistema, pues siempre se entrega un bit de más al inicio que es descartado por el decodificador.

Figura B.15. Diagrama final del decodificador Manchester



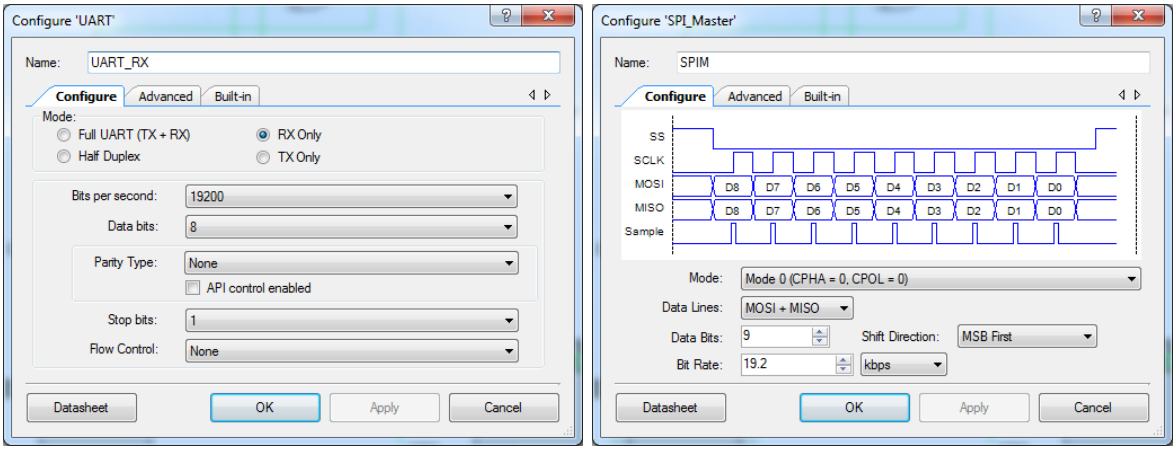
Interfaz con el módulo XBee-Pro

Se implemento una interfaz que permita la comunicación entre la tarjeta de desarrollo y el módulo RF. Se debe tener en cuenta que el módulo XBee-Pro entrega la información a 19200 bps en el formato empleado por el protocolo UART (estado de inactividad alto y bit de inicio y fin de transmisión), y debe ser entregado al decodificador con estado de inactividad bajo y sin bits de inicio y fin.

La interfaz consta de un componente UART receptor y un componente SPIM conectados como se muestra en la Figura B.14 y los parámetros configurados como en la Figura B.15.

Esta interfaz recibe la cadena de bits mediante el módulo UART y por medio de código se envía el dato capturado a través de un pin de salida conectado a la salida *mosi* del SPIM.

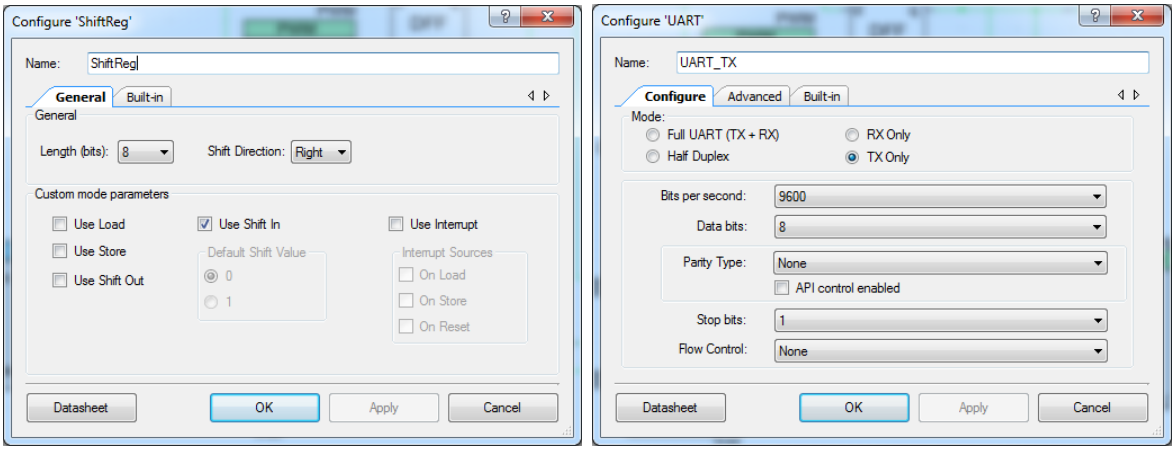
Figura B.16. Configuración UART y SPIM



Interfaz con el Computador

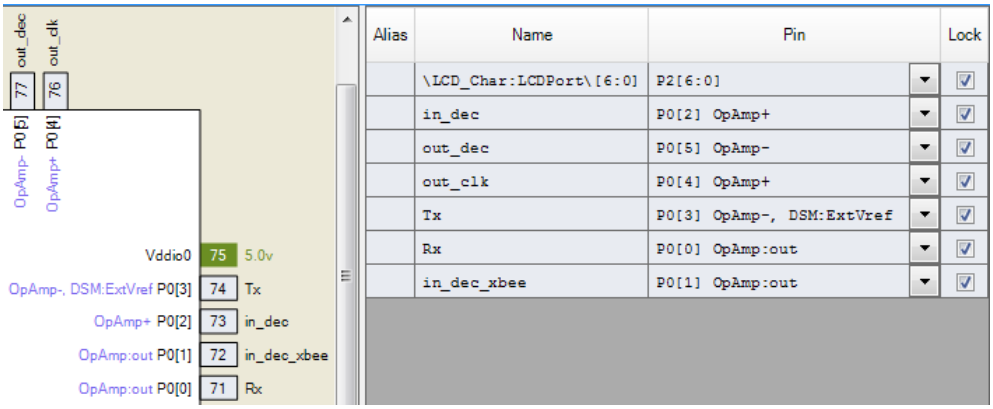
La conexión con el computador portátil consta de un módulo Shift Register y un UART conectados como se observa en la Figura B.14 y configurados como en la Figura B.16. Esta interfaz captura la información saliente del decodificador (dato decodificado), en el Shift Register. Esta interfaz hace uso de la señal de reloj recuperada en el decodificador para activar el almacenamiento de la información. Luego por medio de código cada Byte de datos recuperado es enviado mediante el módulo UART al pin de transmisión serie de la interfaz RS232 de la tarjeta de desarrollo, a donde va conectado el cable RS232 y a este el computador portátil.

Figura B.17. Configuración Shift Register y UART



Seguido de esto se hace la configuración de pines para terminar la configuración hardware del proyecto, los pines se establecen como en la Figura B.17. Se utilizó el puerto 0 para las entradas y salidas, y el puerto 2 para la conexión del LCD utilizado para el monitoreo del sistema.

Figura B.18. Configuración de pines para el decodificador Manchester



Como parte de la codificación software, se crea el código que permite el manejo del decodificador, las funciones utilizadas se encuentran en el Datasheet de cada módulo. En la Tabla B.3 se encuentra la descripción de algunas de las más importantes.

Tabla B.3. Funciones utilizadas en el decodificador Manchester

Módulo	Función	Descripción
SPIM	SPIM_Start()	Se debe llamar en el inicio para inicializar el componente.
SPIM	SPIM_WriteByte()	Coloca un byte en el buffer de transmisión que se enviará en el siguiente intervalo de tiempo disponible.
ShiftReg	ShiftReg_Start()	Inicial el componente y habilita todas las interrupciones seleccionadas
ShiftReg	ShiftReg_ReadRegValue()	Lee el valor actual del registro de desplazamiento
ShiftReg	ShiftReg_GetFIFOStatus ()	Retorna el actual estado de la entrada o salida FIFO
UART	UART_Start()	Inicializa y habilita la operación del UART
UART	UART_ReadRxStatus()	Retorna el actual estado del registro Status de recepción
UART	UART_ReadRxData()	Retorna el dato en el registro de dato de recepción
UART	UART_PutArray()	Trae los datos desde una cadena de memoria y los ubica dentro del buffer de memoria para transmitirlos
PWM	PWM_Start()	Inicializa el PWM con los valores personalizados por defecto

La aplicación está constantemente verificando si hay algún dato disponible en el buffer del UART receptor, cuando lo hay, toma los dos bytes que representan los 8 bits más significativos y menos significativos de un byte codificado. Por medio de una rutina para evitar errores con el primer bit, se envían estos bytes a la entrada del decodificador. Mientras tanto, el Shift Register se encuentra a la espera de obtener un dato decodificado a la salida del decodificador, cuando lo hay, se captura y se envía por medio del UART transmisor a la sección RS323 de la tarjeta de desarrollo.

El código se muestra e continuación.

```
/* ========================================================================================
 * DECODIFICADOR MANCHESTER
 * ========================================================================================*/
#include <device.h>

void main()
{
uint8 dato;            //Declaración de variables
uint8 dato_rx[2];
LCD_Char_Start();      //Inicialización de módulos utilizados
ShiftReg_Start();
SPIM_Start();
PWM_Start();
UART_TX_Start();
UART_RX_Start();
CYGlobalIntEnable;     //Habilita interrupciones globales
LCD_Char_PrintString("ini");

while(1)               //bucle infinito
       {
       if(UART_RX_ReadRxStatus() & UART_RX_RX_STS_FIFO_NOTEMPTY) //Revisa si se recibe dato
                                                                  en UART RX
              {
              dato_rx[0]=UART_RX_ReadRxData();//Almacena los 8 bits menos significativos de
                                              dato codificado
              CyDelay(1);
              dato_rx[1]=UART_RX_ReadRxData();//Almacena 8 bits más significativos de dato
                                              codificado
              CyDelay(1);
              if(dato_rx[1]>127)//Rutina que envía dato a la entrada del decodificador sin
                                error en el primer bit
                     {
                     SPIM_WriteByte(dato_rx[0]+256);
                     SPIM_WriteByte(dato_rx[1]+256);
                     }
              else
                     {
                     SPIM_WriteByte(dato_rx[0]+256);
                     SPIM_WriteByte(dato_rx[1]);
                     }
              while(ShiftReg_GetFIFOStatus(ShiftReg_IN_FIFO)==ShiftReg_RET_FIFO_EMPTY)
              //Revisa si hay dato decodificado a la salida del decodificador
                     {
                     }
              CyDelay(1);
              dato=ShiftReg_ReadRegValue(); //Captura del dato decodificado
              if(dato!=0)
                     {
                     UART_TX_PutChar(dato);//Envió del dato decodificado al computador
                     LCD_Char_WriteData(dato);
                     }
              }
       }
}
/*========================================================================*/
```

Por último, en la Tabla B.4 se detalla la correspondencia de pines para conectar el XBee-Pro con la tarjeta y la tarjeta con el cable hacia el computador. Adicionalmente se debe hace las conexiones internas que se establecen en la Tabla B.5: in_dec_xbee=P0[1] con in_dec=P0[2] y Tx=P0[3] con TX=P16 del módulo RS232 de la tarjeta.ecen

Tabla B.4. Conexión de la Tarjeta de desarrollo con XBee-Pro

Pin en tarjeta de desarrollo	Pin en el Xbee-Pro
Rx=P0[0]	DOUT=3

Tabla B.5. Conexiones internas en la Tarjeta de desarrollo

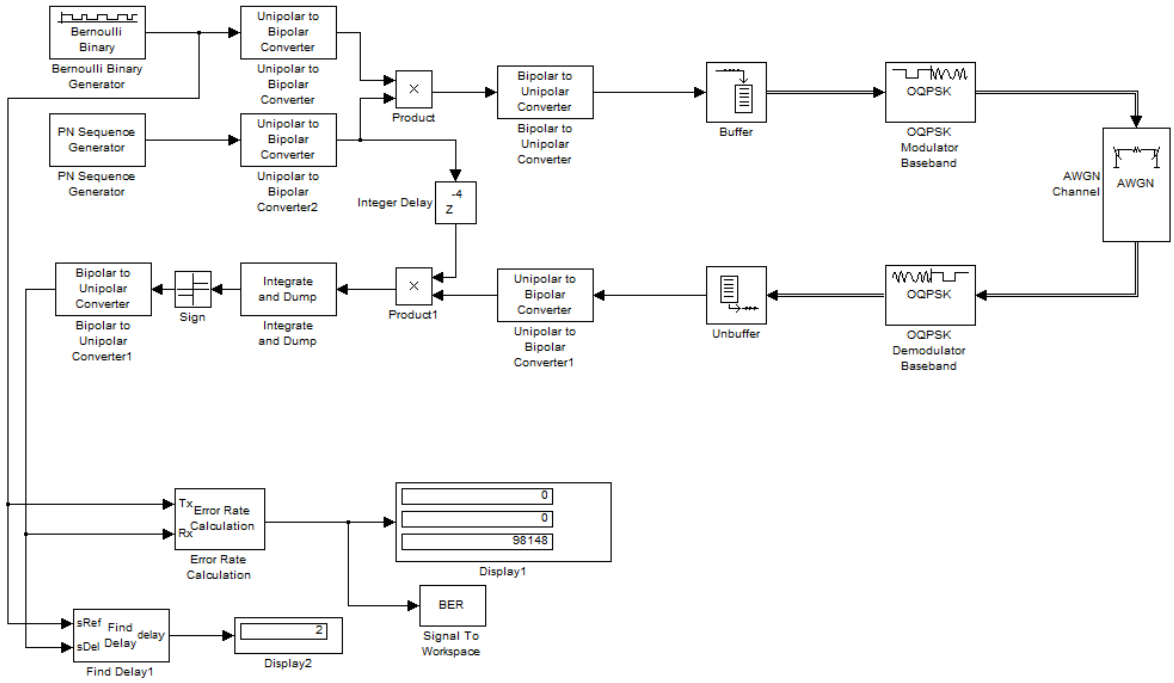
Pin origen	Pin destino
in_dec_xbee=P0[1]	in_dec=P0[2]
Tx=P0[3]	TX=P16 módulo RS232

ANEXO C

DESCRIPCIÓN DE LA SIMULACIÓN DEL SISTEMA DE COMUNICACIÓN

La simulación se realiza con el fin de tener una referencia teórica acerca del desempeño del sistema de comunicación que utiliza el estándar IEEE 802.15.4. Para esto se utiliza la herramienta Simulink®, donde se representa el sistema por medio de bloques que simulan su comportamiento en los que se destacan el ensanchamiento y desensanchamiento de la información, la modulación y demodulación OQPSK y en canal AWGN. Para visualizar el desempeño se utilizan bloques que despliegan resultados obtenidos al finalizar la simulación y permiten la interacción con la herramienta BerTool[4]. El diagrama de bloques del sistema realizado en Simulink se muestra en la Figura C.1.

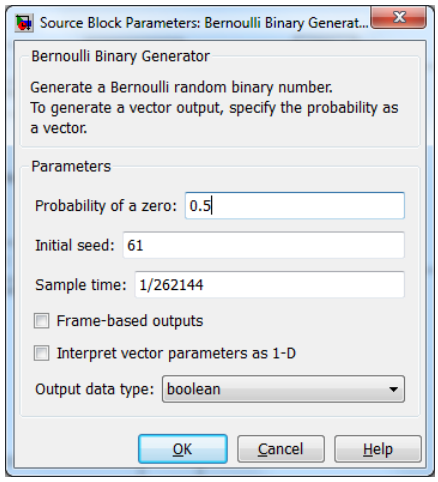
Figura C.1. Diagrama de Bloques de Simulación



- Generador Binario de Bernoulli (*Bernoulli Binary Generator*): Este bloque es el encargado de generar los bits de información que se desean transmitir. Como se transmite una señal a 250 Kbps el tiempo de muestreo (*simple time*) se igual a 1/262144, con la misma probabilidad de generar símbolos de 0 y 1, como muestra la figura C.2.

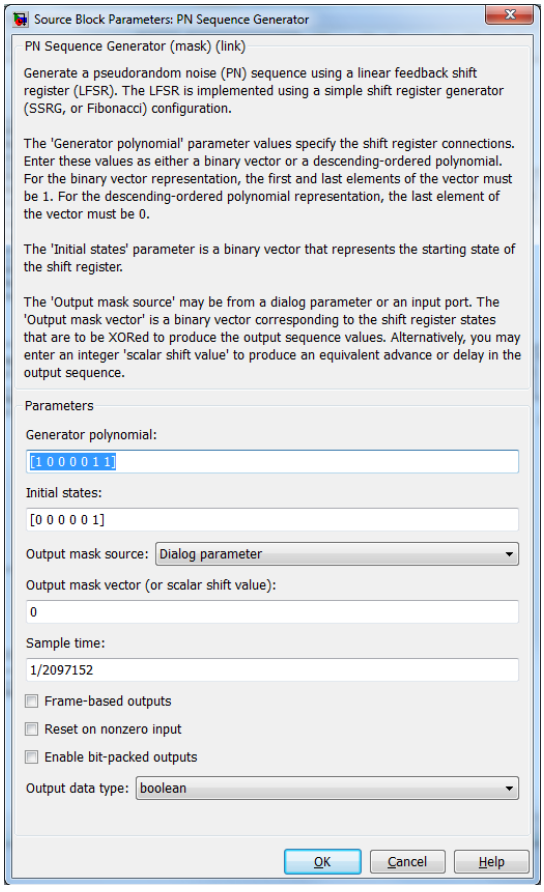
[4] BERTool es una herramienta de interfaz grafica para el análisis del desempeño a nivel de BER y E_b/N_0 en MATLAB

Figura C.2. Generador Binario de Bernoulli



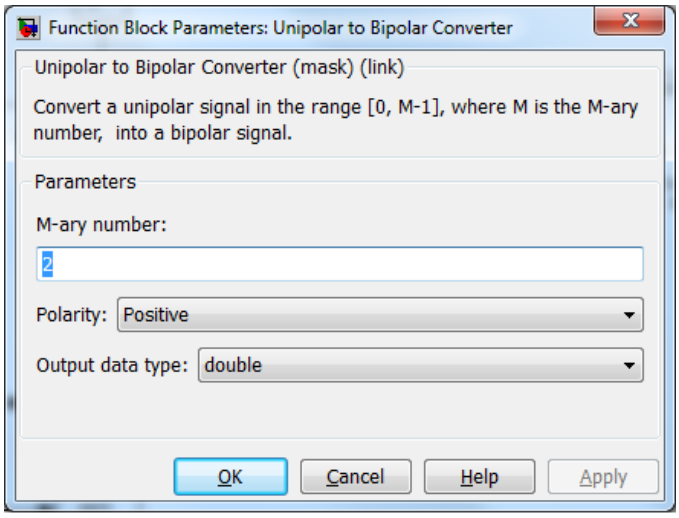
- Generador de Secuencias Pseudo-aleatorias (*PN sequence Generator*): Este bloque es el encargado de generar la secuencia pseudo-aletoria usada en el proceso de ensanchamiento y de desensanchamiento de la información. El tiempo de muestreo para la generación de la secuencia pseudo-aleatoria es de 2 Mbps, como se muestra en la Figura C.3.

Figura C.3. Generador de Secuencias Pseudo-aleatorias



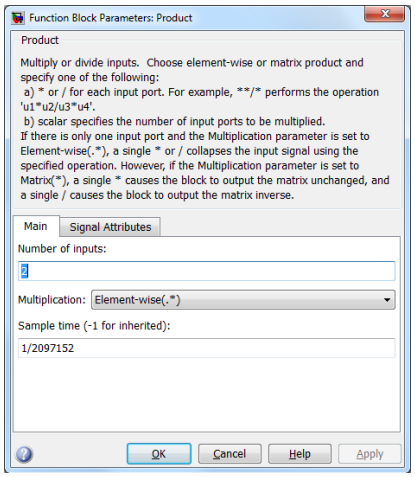
- Conversor de Unipolar a Bipolar (*Unipolar to Bipolar Converter*): Este bloque es el encargado de asignar valores positivos y negativos normalizados a las señales binarias con el objetivo de facilitar el proceso de multiplicación entre la señal de datos y la secuencia pseudo-aleatoria.

Figura C.4. Conversor Unipolar a Bipolar



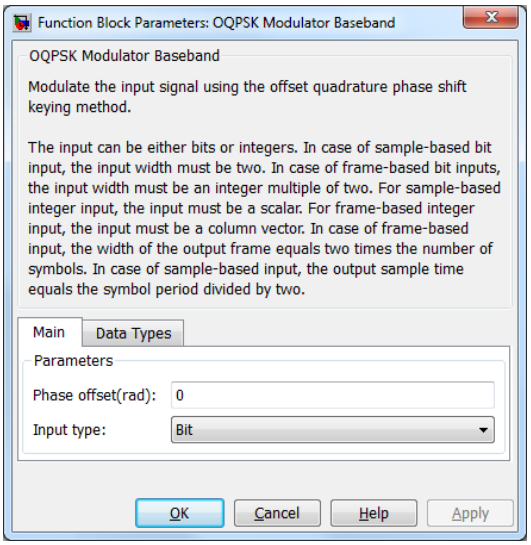
- Producto (*Product*): Este bloque es el encargado de hacer el ensanchamiento y desensanchamiento de los datos al multiplicar las señales correspondientes a los datos y la secuencia pseudo-aleatoria. Para garantizar que la señal tenga la misma velocidad que la secuencia de ensanchamiento el tiempo de muestreo es el usado en el bloque que genera la secuencia pseudo-aleatoria.

Figura C.5. Producto



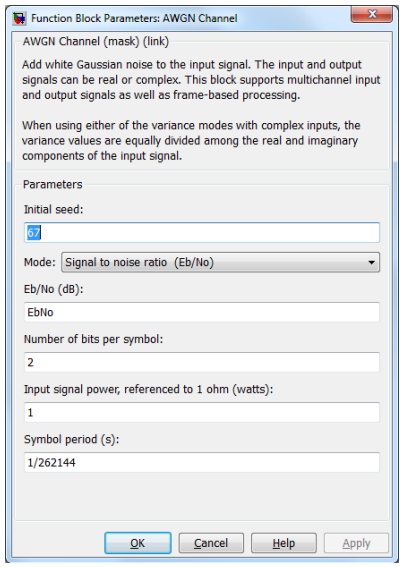
- Modulador OQPSK (*OQPSK Modulator Baseband*): Este bloque es el encargado de realizar la modulación OQPSK, donde el tipo de entrada de datos es Bit y no se hace un desplazamiento de fase adicional al que ya tiene la modulación.

Figura C.6. Modulador OQPSK



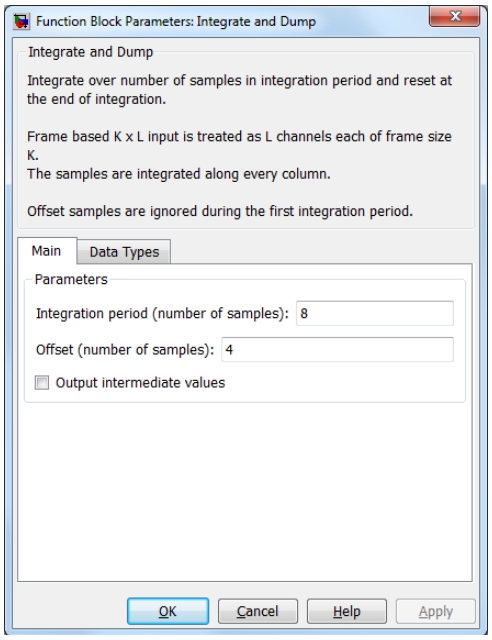
- Canal AWGN (*AWGN Channel*): Este bloque representa el comportamiento de un canal AWGN, a partir de las variaciones de la relación E_b/N_0. Como la señal de entrada posee una velocidad de 2 Mbps (Velocidad del chip), se ha definido un periodo de símbolo igual a 1/262144, correspondiente a una velocidad de bits igual a 254 Kbps. Además, como la señal se modula OQPSK el numero de bits por símbolo son 2.

Figura C.7. Canal AWGN



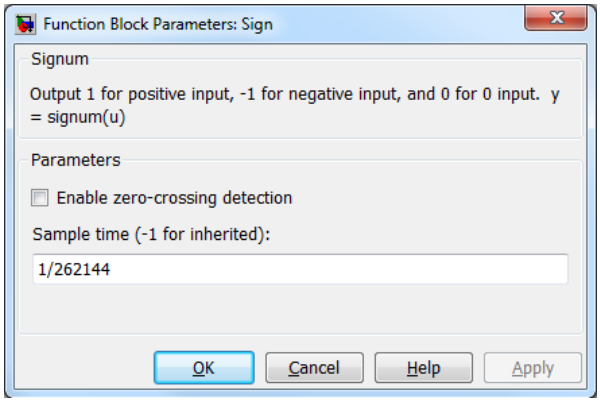
- Integración y Descarga (*Integrate and dump*): Este bloque es el encargado de correlacionar la información recibida de acuerdo a las secuencias pseudo-aleatorias. El periodo de integración corresponde al factor de ensanchamiento, que para el sistema es 8.

Figura C.8. Integración y Descarga



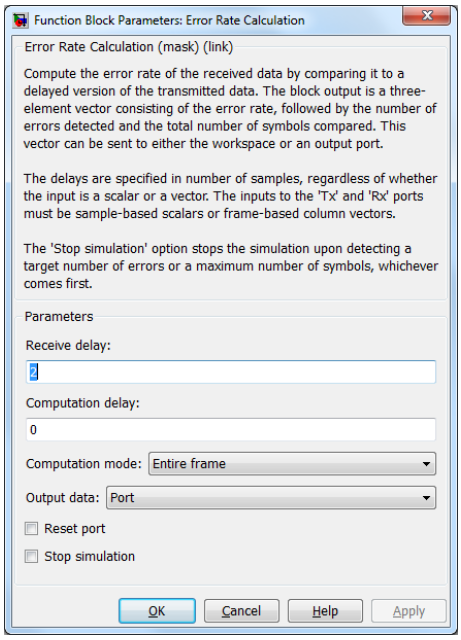
- Signo (*Sign*): Este bloque da como resultado a un valor igual a 1 si a la entrada se presenta un valor positivo, mientras que si se tiene a la entrada un valor negativo, la salida es -1. El tiempo de muestreo corresponde a 256 Kbps.

Figura C.9. Signo



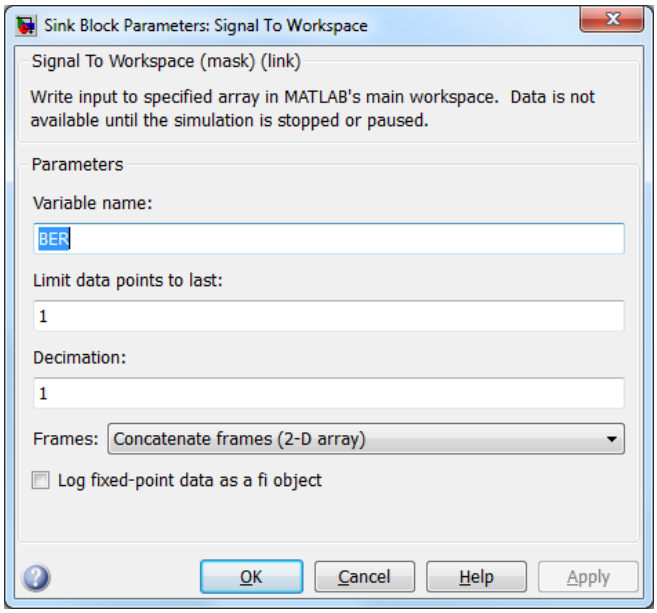
- Calculo de Tasa de Errores (*Error Rate Calculation*): Este bloque es el encargado de realizar la comparación entre cada uno de los bits recibidos y los bits transmitidos. Para que esta comparación sea correcta las dos señales deben estar totalmente sincronizadas.

Figura C.10. Calculo de la Tasa de Errores



- Señal al Espacio de Trabajo (*Signal to Workspace*): este bloque es el encargado de enviar los datos correspondientes a BER hacia un espacio de trabajo exterior, para que sean tomados por BERTool, obtenidos como resultado de la simulación al modificar la variable E_b/N_0 descriota en el canal AWGN. En el bloque de la Figura C.11 es necesario establecer solamente el nombre de la variable, para que el Workspace la interprete para visualizar el desempeño del sistema.

Figura C.11. Señal al Espacio de Trabajo



REFERENCIAS BIBLIOGRÁFICAS

[1] Cypress Semiconductor,“Manchester Encoder Using PSoC™”, Cypress Semiconductor Corporation, United States of America, 2005.

[2] http://www.cypress.com/?id=2494&source=home_documentation [Consultado: Octubre 2011].

[3] Cypress Semiconductor,“Serial Peripheral Interface (SPI) Master 2.20”, Cypress Semiconductor Corporation, United States of America, 2005.

[4] Cypress Semiconductor,“Shift Register (ShiftReg) 2.0”, Cypress Semiconductor Corporation, United States of America, 2005.

[5] Cypress Semiconductor,“Universal Asynchronous Receiver Transmitter (UART) 2.10”, Cypress Semiconductor Corporation, United States of America, 2005.

[6] Cypress Semiconductor,“Manchester Decoder Using PSoC®”, Cypress Semiconductor Corporation, United States of America, 2007.

[7] http://www.trendnet.com/langsp/products/proddetail.asp?prod=150_TU-S9&cat=32 [Consultado: Julio 2011].