Instituto Tecnológico Autónomo de México

Departamento Académico de Sistemas Digitales

*Nuevas Tecnologías Inalámbricas*

***Práctica 5***

***Sensores analógicos con Xbee Pro de 900 MHz***

Integrantes:

Alfonso Venancio- 149211

Efraín Aguilar- 149643

Leandro Pantoja- 150883

Ulises Alejandre - 159235

19 de febrero de 2019

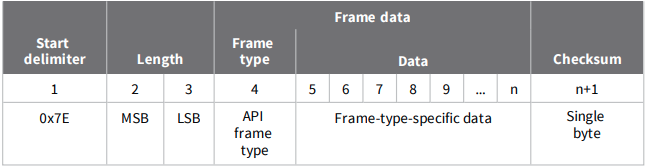
Introducción

Un puerto serial es una interfaz de comunicación asíncrona en el que se transfiere información bit por bit. Las computadoras modernas que no cuentan con puerto serial requieren un convertidor de USB a serial para ser compatibles con dispositivos RS-232.

XBee es una familia de módulos de radio que están basados en el estándar IEEE 802.15.4-2003 que esta diseñado para comunicación punto a punto y estrella a través del aire y usando velocidades de 250 kbit/s. Los radios XBee puedes operar en dos modos: transparente y basado en paquetes. En el modo transparente, la información en el pin de Data IN se transmite directamente al aire sin ninguna modificación y pueden ser dirigidos a una dirección en específico o a múltiples radios y es común el uso de este modo cuando la información no puede tolerar cambios en el formato de la información.

Para el modo basado en paquetes, la información está envuelta en una estructura de paquete que permite direccionamiento, parámetros y feedback de paquetes, incluyendo de sensado remoto y control de entradas y salidas digitales y entradas analogicas.

Los paquetes o tramas en el modo de operación basado en paquetes se ve de la siguiente manera:



El delimitador de inicio indica el inicio de la trama y siempre es igual a 0x7E. Los bytes del tamaño de la trama corresponden los bytes de los datos (sin contar los delimitadores de inicio, del tamaño y el checksum); el primer byte es él más significativo (MSB) y el segundo el menos significativo (LSB). Dentro del campo de datos de la trama se incluyen varios bytes de datos y un byte para saber el tipo de trama e indica la manera en la que están organizados los datos. Por último, el checksum ayuda a probar la integridad de la trama en el receptor.

Desarrollo

Esta práctica tiene como objetivo poder desarrollar un script de python que pueda recibir los datos de un xbee receptor a través del puerto serial de una computadora. Se usaron dos módulos Xbee con una tarjeta Groove cada uno.

Primero se configuraron los xbee mediante XCTU para que pudieran recibir y transmitir datos analógicos: se les puso un mismo ID de red (0x1234) y se les asignó el modo de operación API [1]; al transmisor se le asignó la dirección MAC del receptor como dirección de destino; se activó el pin DIO3/AD3, que está conectado al potenciómetro como ADC [2]; se le puso un tiempo de espera de 7.5 segundos (0x1D4C) entre envío de muestras y se puso en modo de descanso síncrono-cíclico; al receptor, modo de apoyo de descanso y designado como el coordinador de los descansos (campo SO con 001); se pone en modo de descanso en intervalos de 7.5 segundos como el transmisor (campo SP con 2EE) y se le asigna un periodo de inactividad de 2.5 segundos (9C4 en campo ST). Los demás parámetros se dejaron en su valor por defecto.

Ya que estuvieron configurados, se inició la consola de XCTU para poder observar las tramas que se enviadas. La siguientes figura muestran las tramas del transmisor y el receptor en la consola de XCTU:

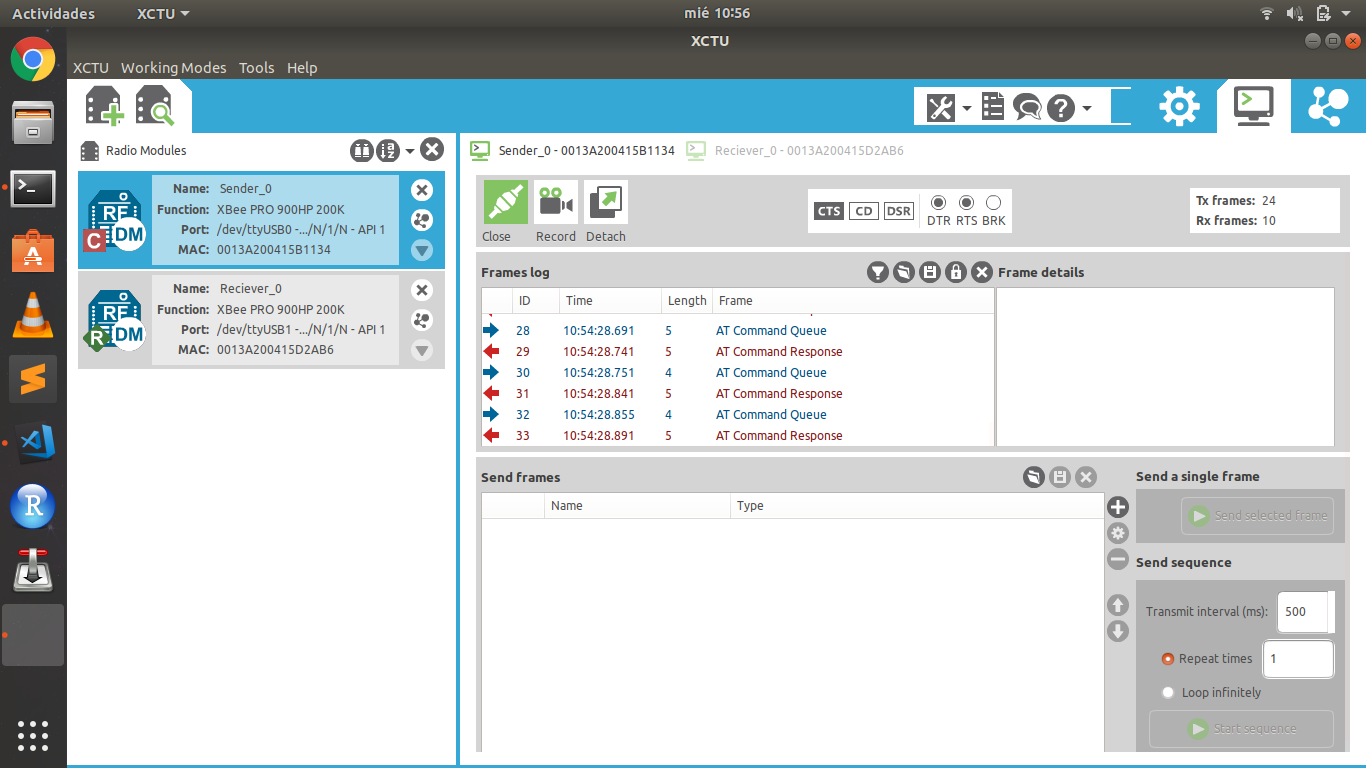
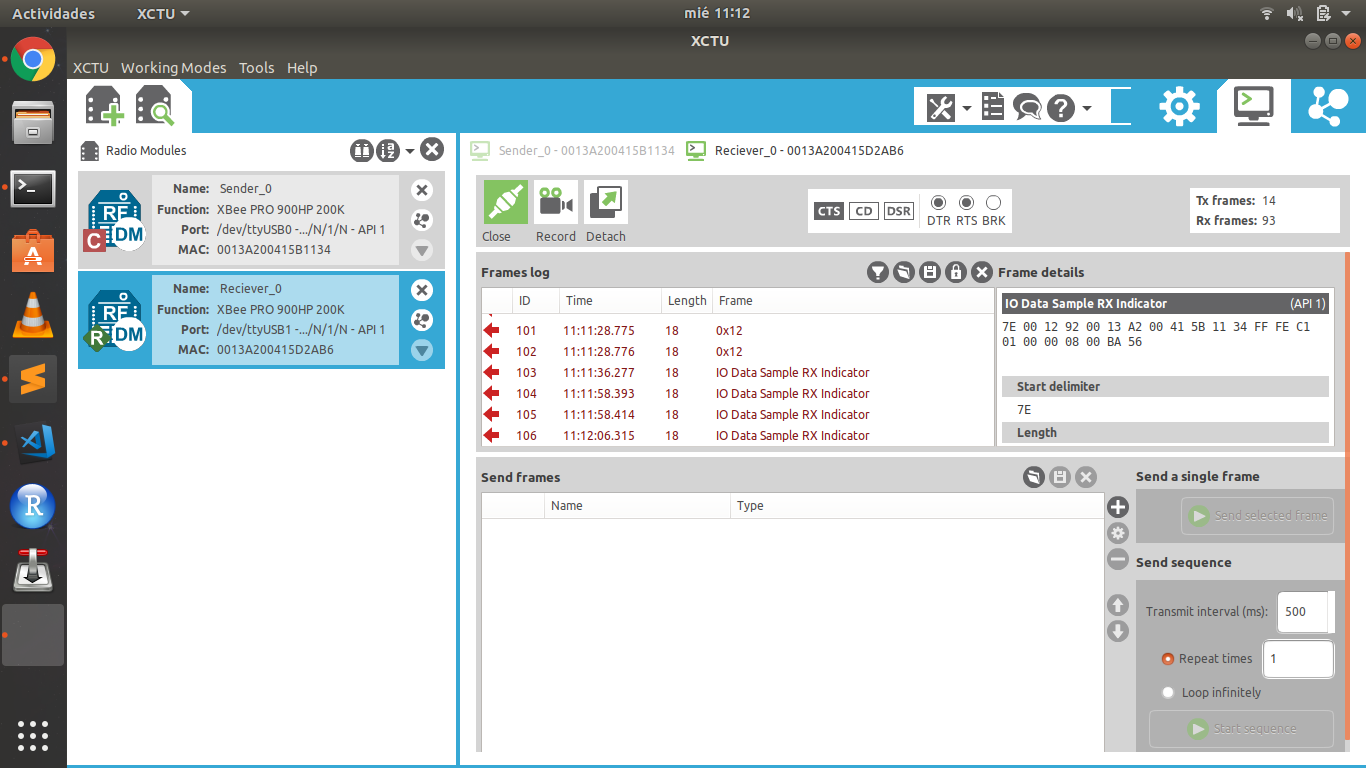
Figura 1. Tramas enviadas

Figura 2. Tramas recibidas

En el campo frame details podemos copiar la información de las tramas a un archivo de texto y nos dió la siguiente información para el transmisor:

*AT Command Response (API 1)*

*7E 00 05 88 90 41 43 00 63*

*Start delimiter: 7E*

*Length: 00 05 (5)*

*Frame type: 88 (AT Command Response)*

*Frame ID: 90 (144)*

*AT Command: 41 43 (AC)*

*Status: 00 (Status OK)*

*Checksum: 63*

Para el receptor obtuvimos lo siguiente:

*IO Data Sample RX Indicator (API 1)*

*7E 00 12 92 00 13 A2 00 41 5B 11 34 FF FE C1 01 00 00 08 00 BA 56*

*Start delimiter: 7E*

*Length: 00 12 (18)*

*Frame type: 92 (IO Data Sample RX Indicator)*

*64-bit source address: 00 13 A2 00 41 5B 11 34*

*16-bit source address: FF FE*

*Receive options: C1*

*Number of samples: 01*

*Digital channel mask: 00 00*

*Analog channel mask: 08*

*DIO3/AD3 analog value: 00 BA (186)*

*Checksum: 56*

En ambos casos podemos ver la trama separada en bytes con sus valores en hexadecimal y el significado de cada segmento. En el caso del receptor, el valor que queremos recuperar es el campo de *DIO3/AD3 analog value* con un valor de 0x00ba (186 en base 10). El rango de valores del sensor está discretizado en 2^16 (65,536) niveles. Con esta información escribimos el programa de python que obtiene y muestra la información.

Resultados

Usando un script de python obtuvimos la información del puerto serial al que estaba conectado el Xbee. Se usó la librería *serial* para poder trabajar con el puerto serial y la decimal para poder mostrar el valor de voltaje en un formato más legible (notación científica). El algoritmo del programa es el siguiente

1. Abrir el puerto serial
2. Leer 22 bytes del puerto serial
3. Obtener los valores deseados de la trama de bytes obtenida
4. Convertir el valor de los dos bytes de hexadecimal a decimal
5. Convertir el valor decimal al equivalente en voltaje e imprimirlo en pantalla
6. Cerrar el puerto serial

El código terminado se muestra a continuación:

#python 3.7.1

import serial

from decimal import Decimal

#abrir el puerto serial para el receptor

ser = serial.Serial(port='/dev/ttyUSB1')

#leer la trama de 22 bytes que llegue por el puerto serial

aux = ser.read(22)

#Ejemplos de lecturas del puerto serial

#aux = b'~\x00\x12\x92\x00\x13\xa2\x00A[\x114\xff\xfe\xc1\x01\x00\x00\x08\x00\xbaV'

#aux = b'~\x00\x12\x92\x00\x13\xa2\x00A[\x114\xff\xfe\xc1\x01\x00\x00\x08\x00\xb7Y'

print(str(aux) + ' <- trama del Xbee')

#tomar sólo los valores del sensor

x = aux[-3:-1]

print(str(x) + '<- bytes del voltaje')

#reemplazar los símbolos que representan un valor hexadecimal

x2 = str(x).replace("x","")

x3 = str(x2).replace("\\","")

#separar los valores deseados de la trama

valor = x3[-5:-1]

#convertir a decimal

val = int(valor,16)

print(str(val) + ' <- valor decimal de los bytes')

#conversion a volts

Vmax = 3.3 #valor max posible del potenciometro

maxBits = 2\*\*16 #numero de combinaciones de bits posibles

vPot = val\*Vmax/maxBits #voltaje del potenciometro

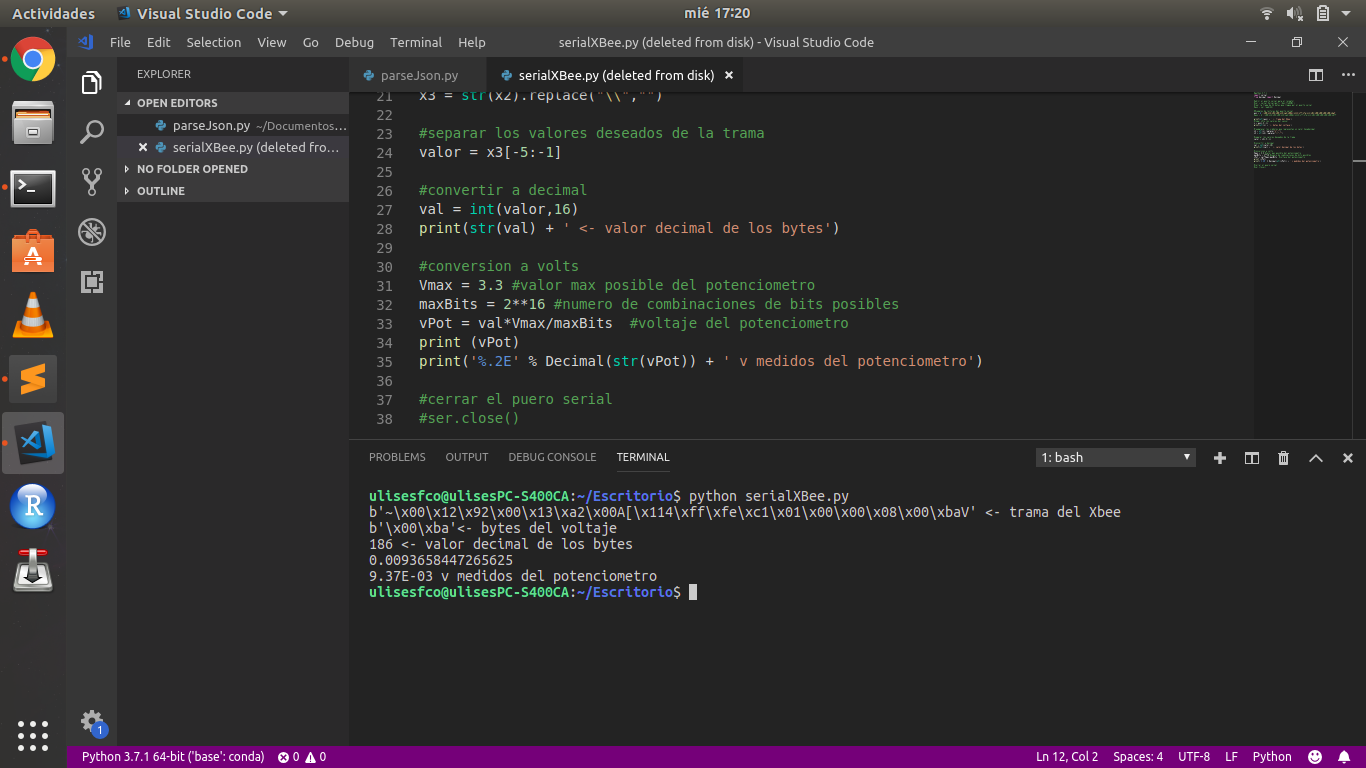
print (vPot)

print('%.2E' % Decimal(str(vPot)) + ' v medidos del potenciometro')

#cerrar el puerto serial

ser.close()

La siguiente imagen muestra el resultado de una medición de voltaje:

Figura 3. Salida del programa de python

Aquí podemos ver la lectura tuvo un valor de 9.37 mV a partir de la siguiente secuencia de bytes obtenida a través del puerto serial: b'~\x00\x12\x92\x00\x13\xa2\x00A[\x114\xff\xfe\xc1\x01\x00\x00\x08\x00\xbaV'

Conclusión

En esta práctica se pudo observar los modos operacionales del Xbee, en particular el basado en paquetes con él cual pudimos experimentar el envío de tramas y el uso de Python para desempaquetarlas y poder decodificar la información transferida de un Xbee a otro. Esto es de gran utilidad porque pudimos observar cómo podría funcionar una red de varios Xbees que manden información de algún tipo a una base de datos.

Referencias

* <https://www.digi.com/resources/documentation/digidocs/pdfs/90001457-13.pdf>
* <https://www.digi.com/resources/documentation/digidocs/pdfs/90001496.pdf>
* <https://pyserial.readthedocs.io/en/latest/shortintro.html>