Creating a Simple Zigbee Communication Network using XBee

ECE-480 SS13 DT2

Outline:

- What is Zigbee?
- Difference between XBee Products
- Introduce Example Project
- Hardware Setup
- Software Setup
  - X-CTU
  - XBee programming
- Collect incoming data using Python
- Summary

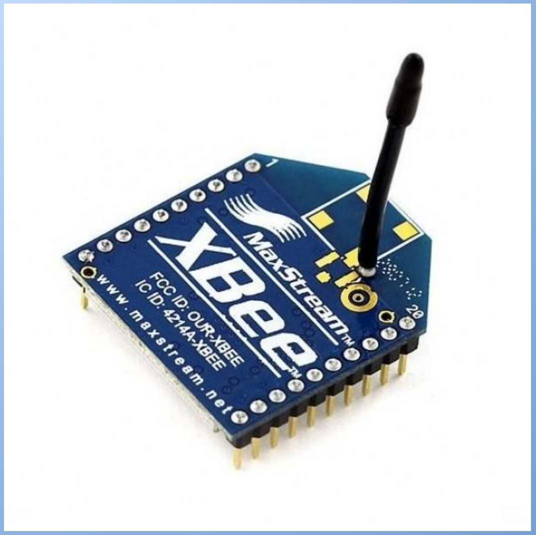
What is Zigbee?

It is a technical standard for communication protocols using small, low power, digital radios for personal area networks (PAN), IEEE International Standard 802.15.4, typically operating at 2.4 GHz.

It's target market is low power applications with infrequent data transmission needs.

What is XBee?

Xbee is Digi International's in house Zigbee communication module brand.



XBee® Family Features Comparison

Protocol	Product	Certified Regions	Frequency	Positioning	RF Line of Sight Range	Transmit Power	Receiver Sensitivity	Form Factor	MSRP	RF Data Rate	Programmable Variant	Hardware
IEEE 802.11	XBee® Wi-Fi	US, CA, EU, AU, JP	2.4 GHz	Low-power serial to Wi-Fi b/g/n	N/A	+16 dBm	-93 to -71 dBm	Through-hole, SMT	$35.00	1 to 72 Mbps	N/A	S6B
IEEE 802.15.4	XBee® 802.15.4	US, CA, EU, AU, BR, JP	2.4 GHz	Low-cost, low-power multipoint	300 ft / 90 m	0 dBm	-92 dBm	Through-hole	$19.00	250 Kbps	N/A	S1
	XBee-PRO® 802.15.4	US, CA, AU, BR	2.4 GHz	Extended-range multipoint	1 mile / 1.6 km	+18 dBm	-100 dBm		$32.00	250 Kbps	N/A	S1
		US, CA, EU, AU, BR, JP	2.4 GHz	International/”J” variant	2500 ft / 1 km	+10 dBm	-100 dBm		$32.00	250 Kbps	N/A	S1
Multipoint Proprietary	XBee-PRO® XSC	US, CA, AU	900 MHz	Long-range multipoint for North America	9 miles / 14.5 km	+24 dBm	-107 to -109 dBm	Through-hole	$39.00	10 Kbps or 20 Kbps	N/A	S3B
	XBee-PRO® 868	EU	868 MHz	Long-range multipoint for Europe	25 miles / 40 km	+25 dBm	-112 dBm		$45.00	24 Kbps	N/A	S5
ZigBee® PRO Feature Set	XBee® ZB SMT	US, CA, EU, AU, BR, JP	2.4 GHz	Surface mount, low-cost, low-power, ZigBee PRO Feature Set, EM357	4000 ft / 1.2 km	+8 dBm	-102 dBm	SMT	$17.50	250 Kbps	32 KB Flash / 2 KB RAM	S2C
	XBee-PRO® ZB SMT	US, CA, AU, BR	2.4 GHz	Extended-range, surface mount, ZigBee PRO Feature Set, EM357	2 miles / 3.2 km	+18 dBm	-101 dBm		$28.50	250 Kbps	32 KB Flash / 2 KB RAM	S2C
	XBee® ZB	US, CA, EU, AU, BR, JP	2.4 GHz	Through-hole, low-cost, low-power, ZigBee PRO Feature Set, EM250	400 ft / 120 m	+3 dBm	-96 dBm	Through-hole	$17.00	250 Kbps	N/A	S2
	XBee-PRO® ZB	US, CA, AU, BR	2.4 GHz	Extended-range, through-hole, ZigBee PRO Feature Set, EM250	2 miles / 3.2 km	+18 dBm	-102 dBm		$28.00	250 Kbps	32 KB Flash / 2 KB RAM	S2B
		US, CA, EU, AU, BR, JP	2.4 GHz	International/”J” variant	5000 ft / 1.5 km	+10 dBm	-102 dBm		$28.00	250 Kbps	32 KB Flash / 2 KB RAM	S2B
ZigBee® Smart Energy Public Profile	XBee® SE	US, CA, EU, AU, BR, JP	2.4 GHz	Low-cost, low-power, ZigBee PRO Feature Set	400 ft / 120 m	+3 dBm	-96 dBm	Through-hole	$17.00	250 Kbps	N/A	S2
	XBee-PRO® SE	US, CA, AU, BR	2.4 GHz	Extended-range ZigBee PRO Feature Set	2 miles / 3.2 km	+18 dBm	-102 dBm		$28.00	250 Kbps	N/A	S2B
		US, CA, EU, AU, BR, JP	2.4 GHz	International/”J” variant	5000 ft / 1.5 km	+10 dBm	-102 dBm		$28.00	250 Kbps	N/A	S2B
DigiMesh® Proprietary	XBee-PRO® 900HP	US, CA, AU, BR	900 MHz	Extended-range peer-to-peer mesh, sleeping routers	9 miles / 14.5 km	+24 dBm	-101 to -110 dBm	Through-hole	$39.00	10 Kbps or 200 Kbps	32 KB Flash / 2 KB RAM	S3B
	XBee® 865/868LP	India, EU	865 MHz or 868 MHz	Low-power RF module for India (865 MHz) or Europe (868 MHz) with DigiMesh	2.5 miles / 4 km	+12 dBm	-101 to -106 dBm	SMT	$23.00	10 Kbps or 80 Kbps	32 KB Flash / 2 KB RAM	S8
	XBee® DigiMesh® 2.4	US, CA, EU, AU, BR, JP	2.4 GHz	Low-cost, low-power peer-to-peer mesh, sleeping routers	300 ft / 90 m	0 dBm	-92 dBm	Through-hole	$19.00	250 Kbps	N/A	S1
	XBee-PRO® DigiMesh® 2.4	US, CA, AU, BR	2.4 GHz	Extended-range peer-to-peer mesh, sleeping routers	1 mile / 1.6 km	+18 dBm	-100 dBm		$32.00	250 Kbps	N/A	S1
		US, CA, EU, AU, BR, JP	2.4 GHz	International/”J” variant	3200 ft / 1 km	+10 dBm	-100 dBm		$32.00	250 Kbps	N/A	S1

S1

S2

S2B

S2C

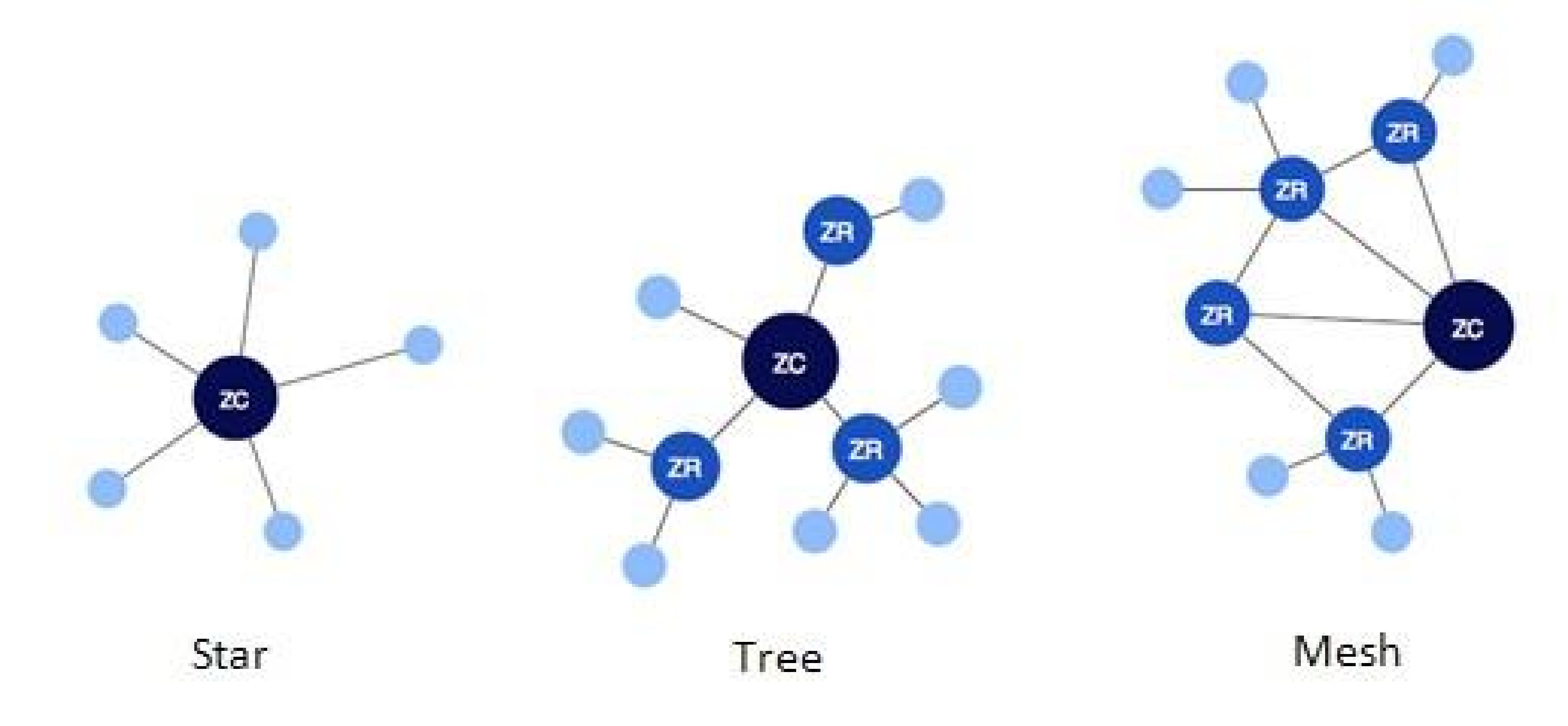
S3B

S5

S6B

S8

Mesh Network Topology

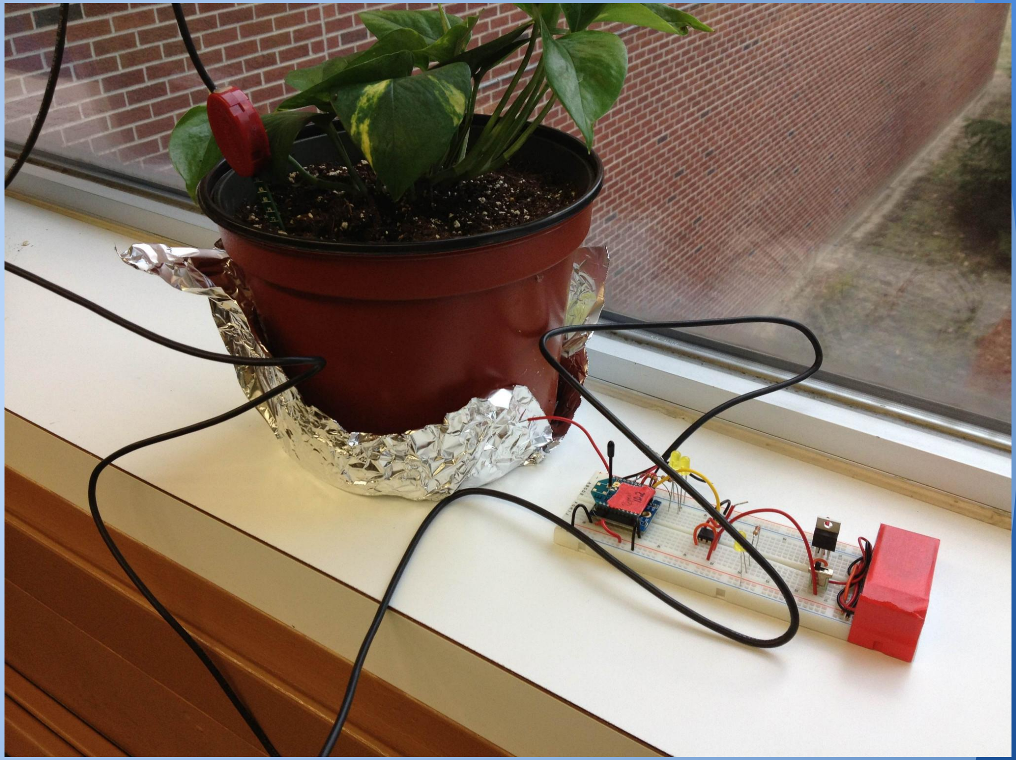


Example Project

- Node connected to moisture sensor that gives off it's reading in volts

- Data transmits to Coordinator (receiver) node

- Data collected using a Python script.



Let's get started. We'll need....

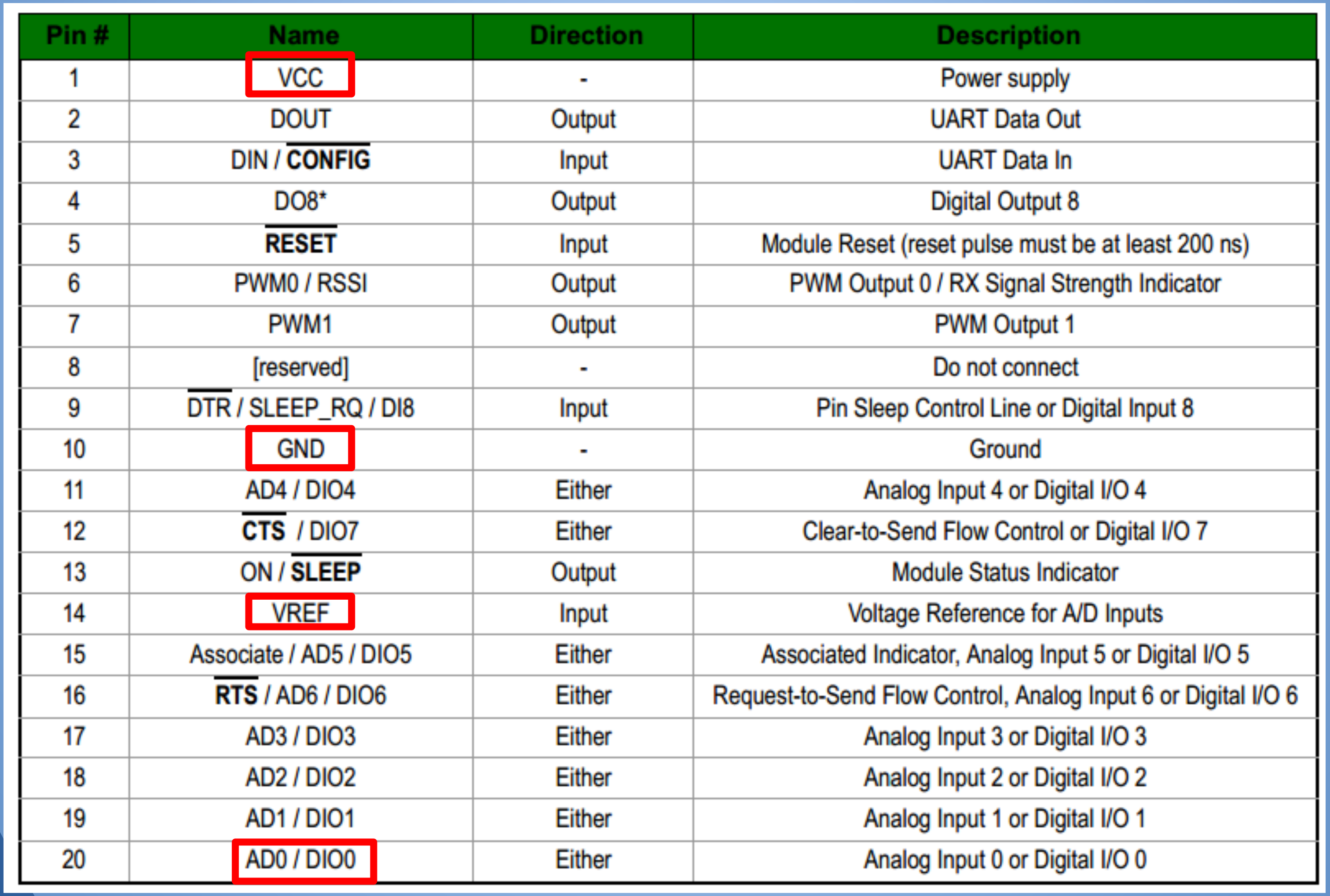
- 2 Xbee DigiMesh 2.4 Units
- Protoboard & Xbee Protoboard Adaptor
- Xbee USB explorer
- X-CTU Tool
- A power supply capable of 3.3V
- Potentiometer/Sensor with Voltage output
- Python
  - Xbee library written for Python
  - pyserial module for Python to interact with your serial port (COM3)

XBee DigiMesh 2.4 RF Module

- DigiMesh Firmware
  - Self healing, ad hoc mesh network
  - Sleep Synchronization
  - All nodes can sleep
- 6 Different 10-bit A/D registers
- Analog Input pins good up to 3.3V
- 90 m range outdoors (with line of sight)

Hardware Setup

🟥 Required



Pin #	Name	Direction	Description
1	VCC	-	Power supply
2	DOUT	Output	UART Data Out
3	DIN / **CONFIG**	Input	UART Data In
4	DO8*	Output	Digital Output 8
5	**RESET**	Input	Module Reset (reset pulse must be at least 200 ns)
6	PWM0 / RSSI	Output	PWM Output 0 / RX Signal Strength Indicator
7	PWM1	Output	PWM Output 1
8	[reserved]	-	Do not connect
9	DTR / SLEEP_RQ / DI8	Input	Pin Sleep Control Line or Digital Input 8
10	GND	-	Ground
11	AD4 / DIO4	Either	Analog Input 4 or Digital I/O 4
12	**CTS** / DIO7	Either	Clear-to-Send Flow Control or Digital I/O 7
13	ON / **SLEEP**	Output	Module Status Indicator
14	VREF	Input	Voltage Reference for A/D Inputs
15	Associate / AD5 / DIO5	Either	Associated Indicator, Analog Input 5 or Digital I/O 5
16	**RTS** / AD6 / DIO6	Either	Request-to-Send Flow Control, Analog Input 6 or Digital I/O 6
17	AD3 / DIO3	Either	Analog Input 3 or Digital I/O 3
18	AD2 / DIO2	Either	Analog Input 2 or Digital I/O 2
19	AD1 / DIO1	Either	Analog Input 1 or Digital I/O 1
20	AD0 / DIO0	Either	Analog Input 0 or Digital I/O 0

Hardware Setup: Power

Zigbee Voltage Supply

- Constraint: 2.8V - 3.4V

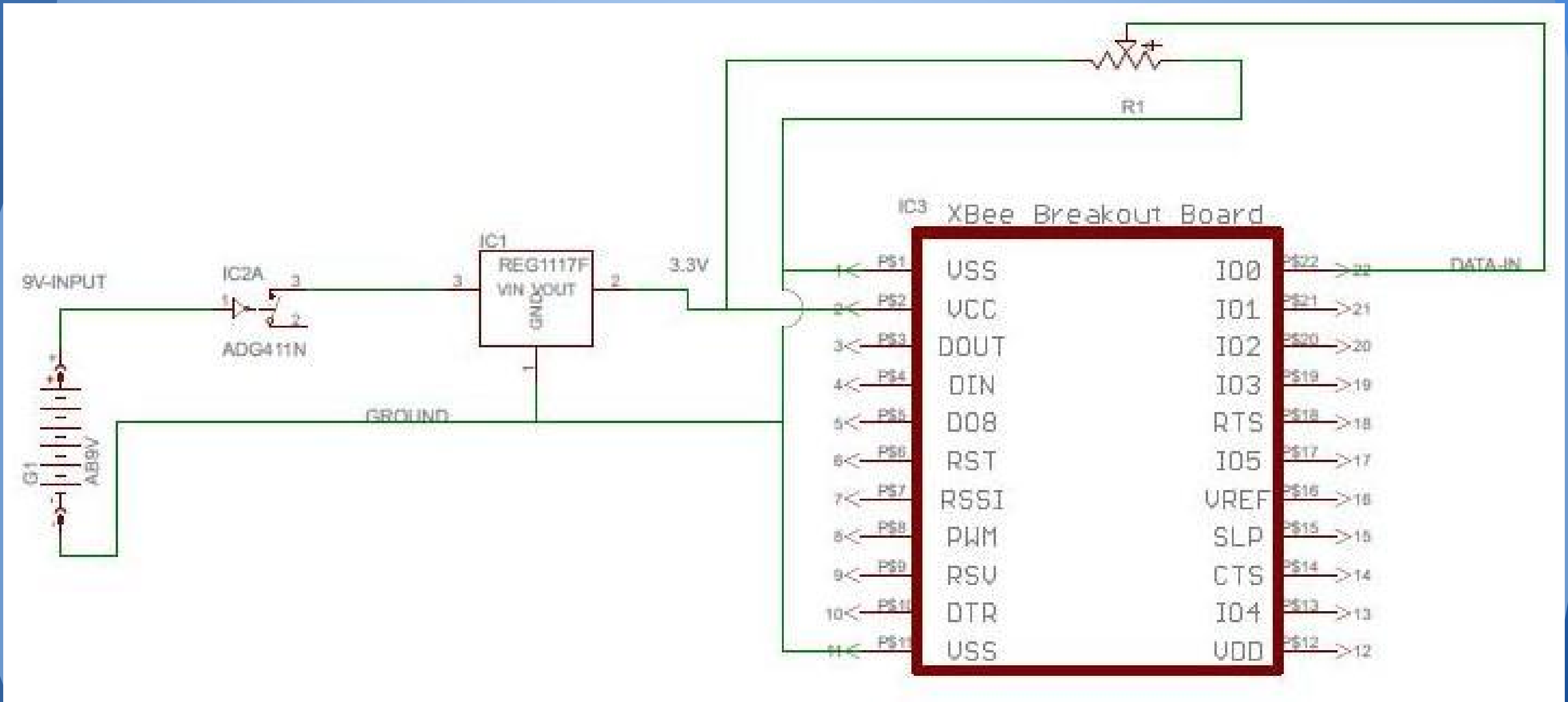
Typical Current Usage

- Idle/Receiving: 50mA
- Transmitting: 45mA
- Powered-down: <50uA

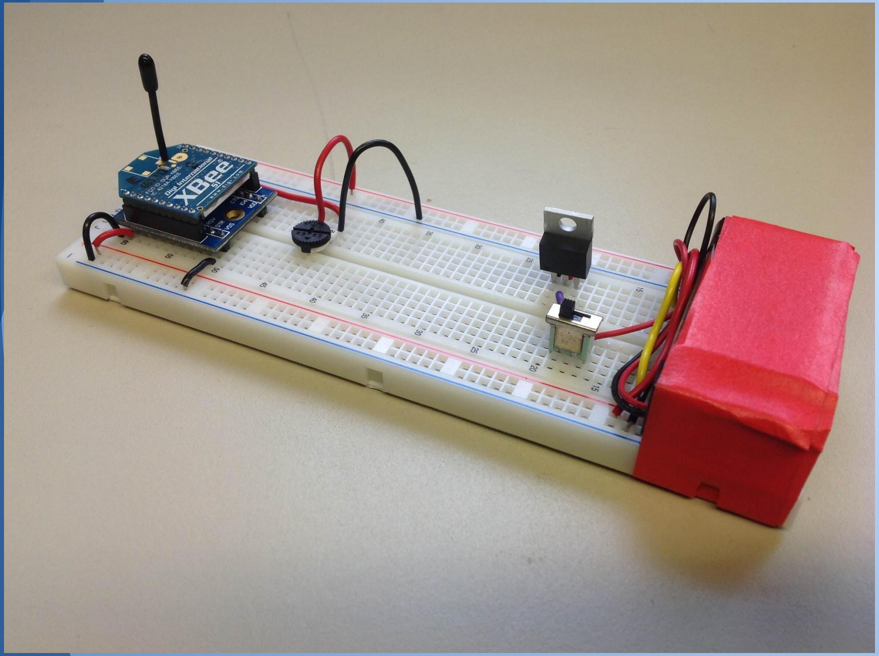
Hardware Setup: Prototype

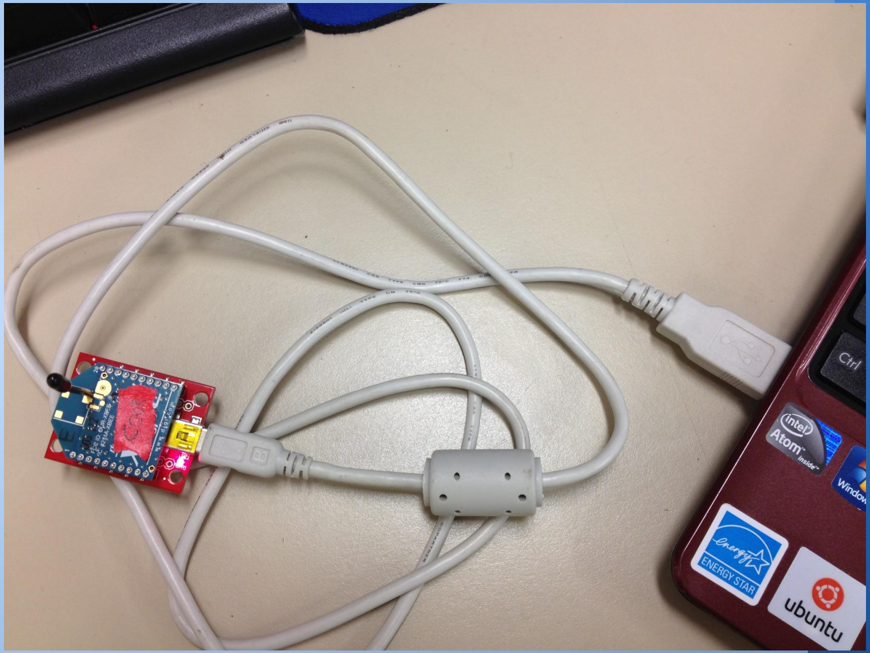
- 3.3V Regulator
- 9V Battery
- XBee Module
- Protoboard & Adapter
- Switch
- Sensor

Hardware Setup: Schematic



Final Hardware Setup

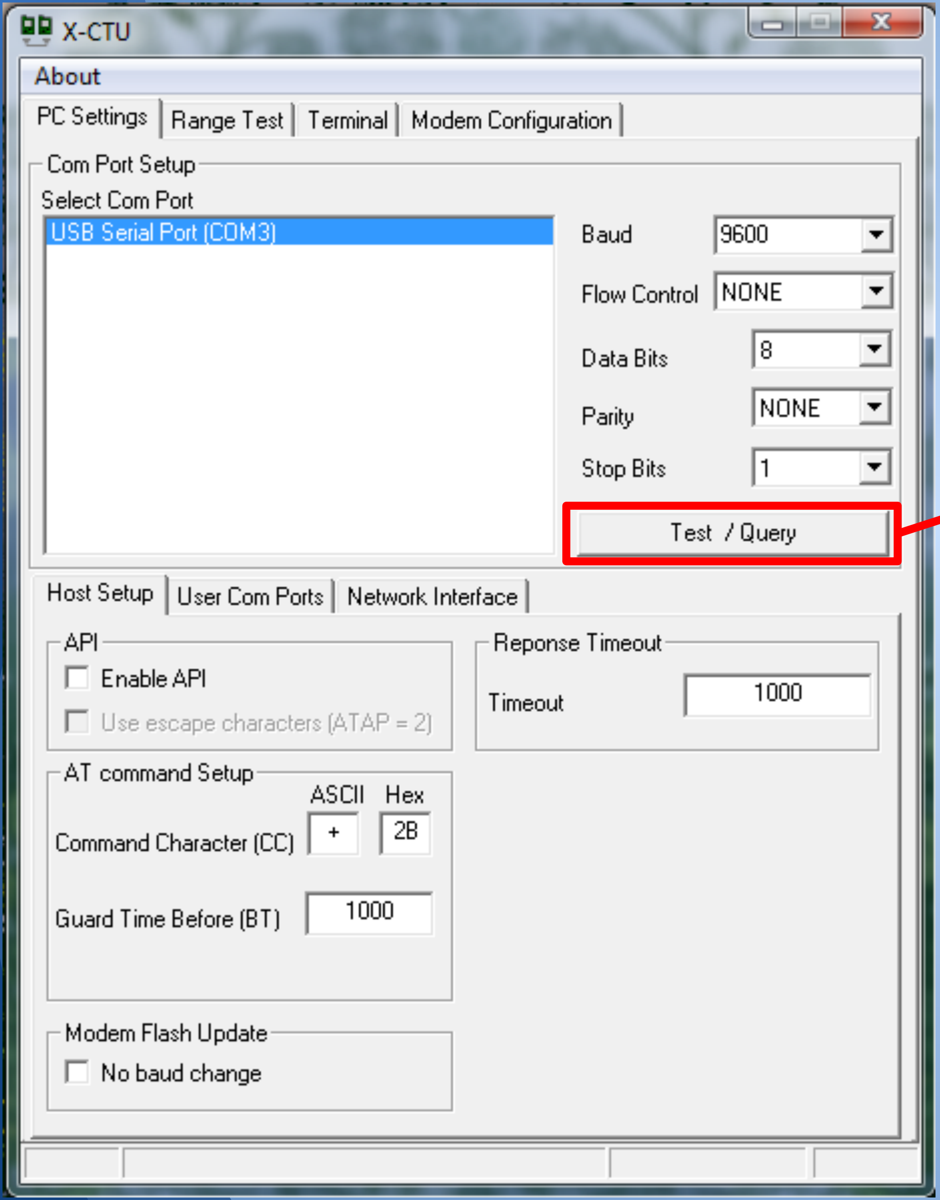




Software: X-CTU

X-CTU is a free software tool available from Digi International to interface with Xbee modules. The tool provides a GUI and terminal interface to configure the modules as well as a built in tool to test the Xbee range and reliability of packet transmissions.

Software: X-CTU



Com Port Setup

Select Com Port

USB Serial Port (COM3)

Baud 9600

Flow Control NONE

Data Bits 8

Parity NONE

Stop Bits 1

API

□ Enable API

□ Use escape characters (ATAP = 2)

Reponse Timeout

Timeout 1000

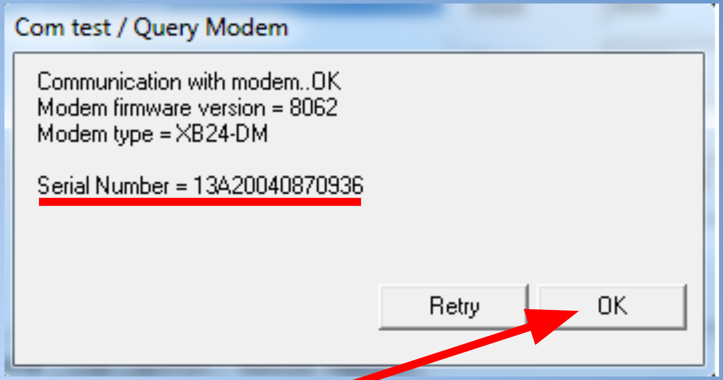
AT command Setup

	ASCII	Hex
Command Character (CC)	+	2B

Guard Time Before (BT) 1000

Modem Flash Update

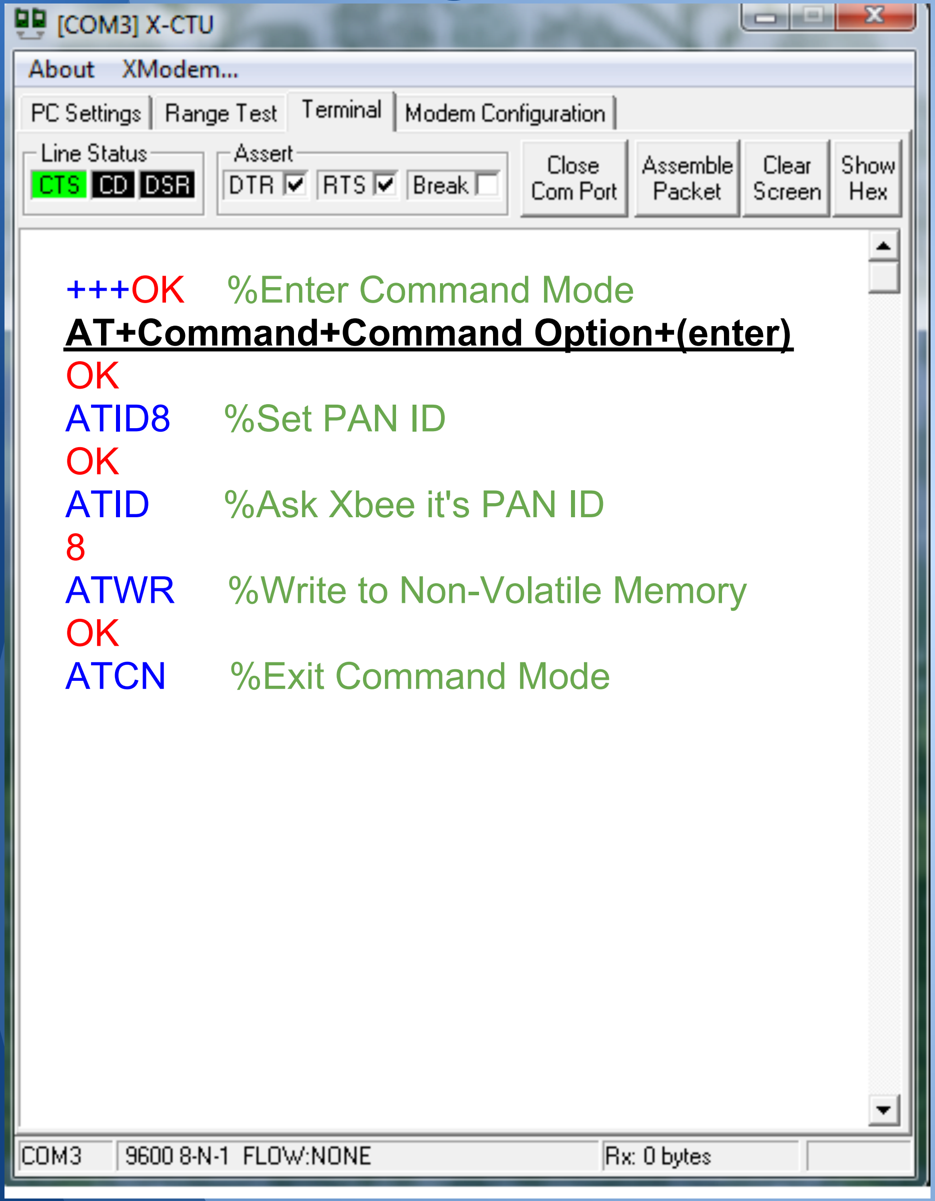
□ No baud change



```
Communication with modem..OK
Modem firmware version = 8062
Modem type = XB24-DM

Serial Number = 13A20040870936
```

- Test Connection
- Note Serial Number

Configuration



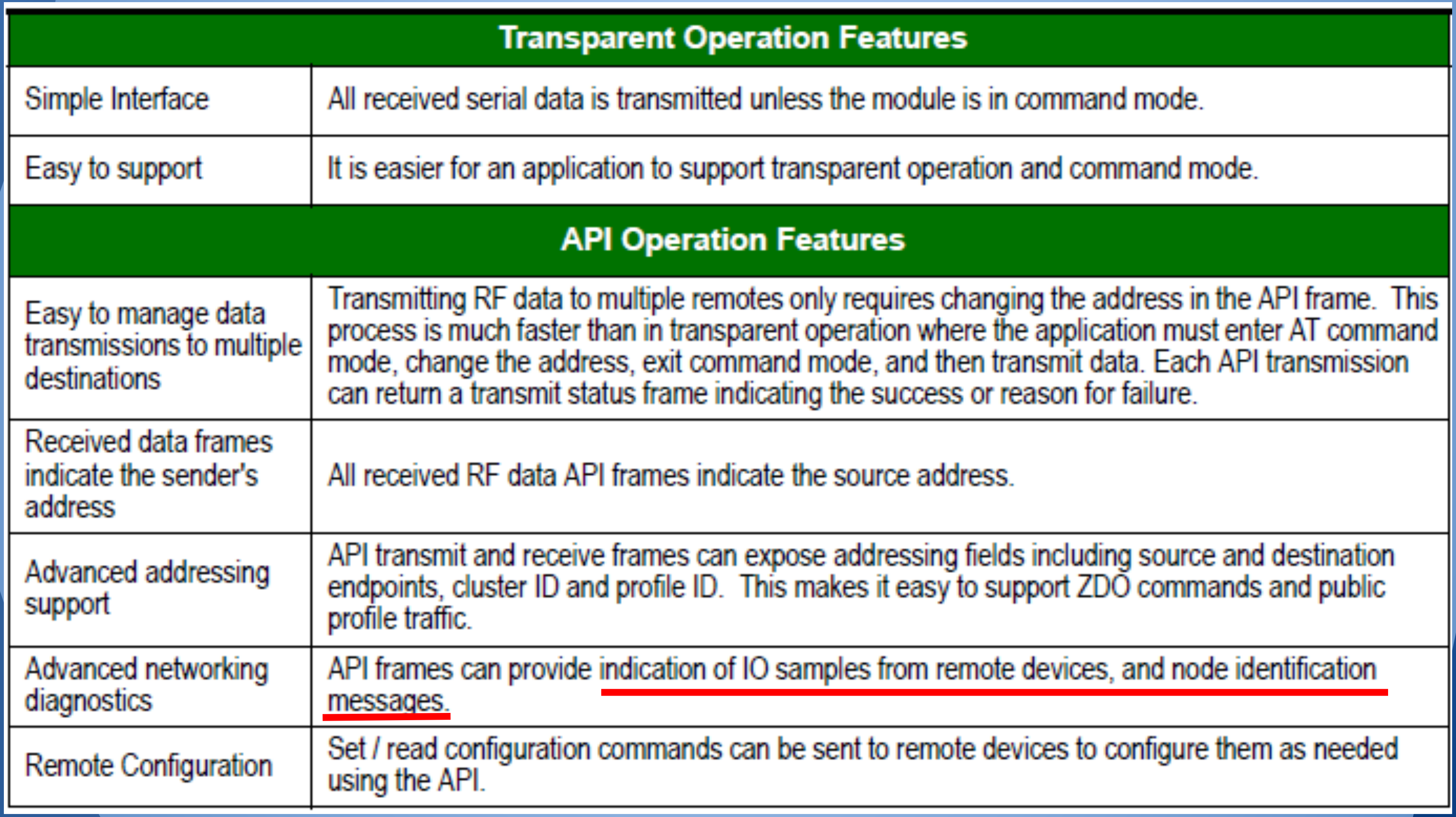
```
+++OK    %Enter Command Mode
AT+Command+Command Option+(enter)
OK
ATID8    %Set PAN ID
OK
ATID     %Ask Xbee it's PAN ID
8
ATWR     %Write to Non-Volatile Memory
OK
ATCN     %Exit Command Mode
```

- Terminal Interface
- +++ : Enter Command Mode
- AT+Command +Command Option +(Enter)

Configuration

- All units will need matching PAN ID, Channel and Sleep Mode settings to function together as one network
- All nodes must have the Coordinators address to know it is the end destination for data transmission
- Nodes must have an analog to digital converter (ADC) enabled and a sample rate set
- Coordinator must be in API mode to see data from node I/O pins

Transparent Mode vs API Mode



Transparent Operation Features	
Simple Interface	All received serial data is transmitted unless the module is in command mode.
Easy to support	It is easier for an application to support transparent operation and command mode.
API Operation Features	
Easy to manage data transmissions to multiple destinations	Transmitting RF data to multiple remotes only requires changing the address in the API frame. This process is much faster than in transparent operation where the application must enter AT command mode, change the address, exit command mode, and then transmit data. Each API transmission can return a transmit status frame indicating the success or reason for failure.
Received data frames indicate the sender's address	All received RF data API frames indicate the source address.
Advanced addressing support	API transmit and receive frames can expose addressing fields including source and destination endpoints, cluster ID and profile ID. This makes it easy to support ZDO commands and public profile traffic.
Advanced networking diagnostics	API frames can provide indication of IO samples from remote devices, and node identification messages.
Remote Configuration	Set / read configuration commands can be sent to remote devices to configure them as needed using the API.

Sleep Mode:

Normal Mode:

- Does not sleep or generate sleep sync messages but will relay sleep sync messages

Cyclic Sleep Mode:

- Will sleep cyclically as determined by the sleep coordinator

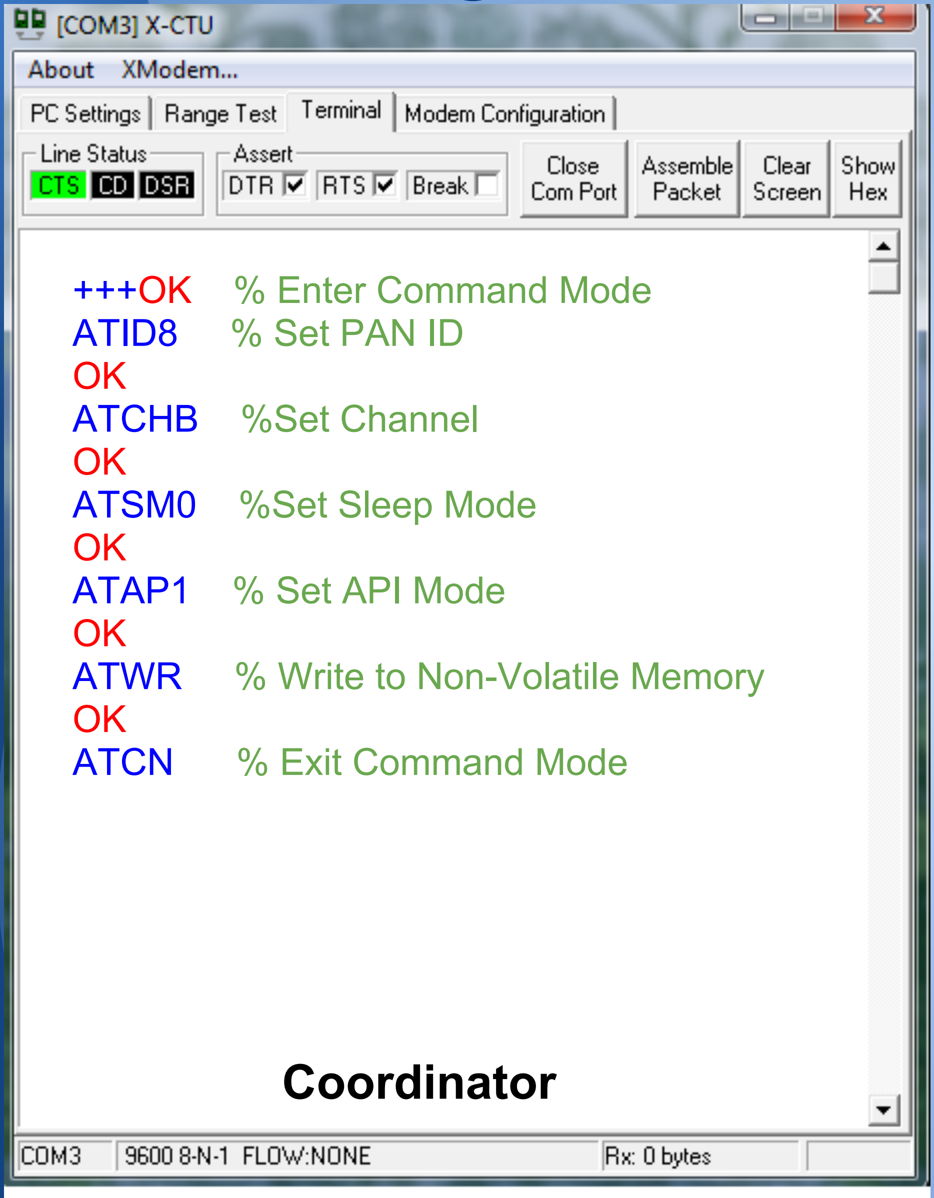
Sleep Support Mode:

- Does not sleep but will generate and relay sleep sync messages

Destination Address:

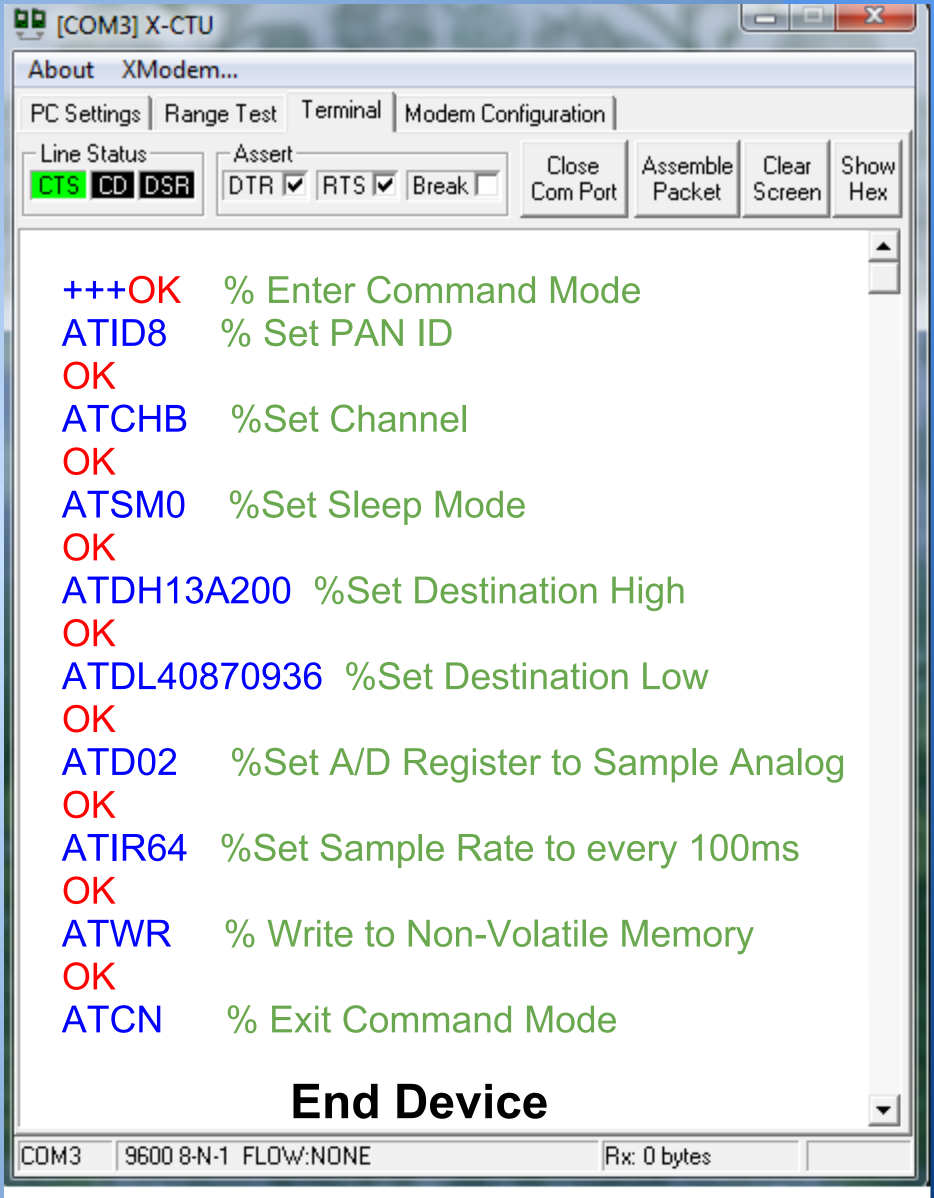
Each XBee has a unique 64-bit serial address that is not changeable by the user, it is printed on the backside of each unit and can also be read off the unit using the X-CTU tool.

Configuration



```
+++OK    % Enter Command Mode
ATID8    % Set PAN ID
OK
ATCHB    %Set Channel
OK
ATSM0    %Set Sleep Mode
OK
ATAP1    % Set API Mode
OK
ATWR     % Write to Non-Volatile Memory
OK
ATCN     % Exit Command Mode
```

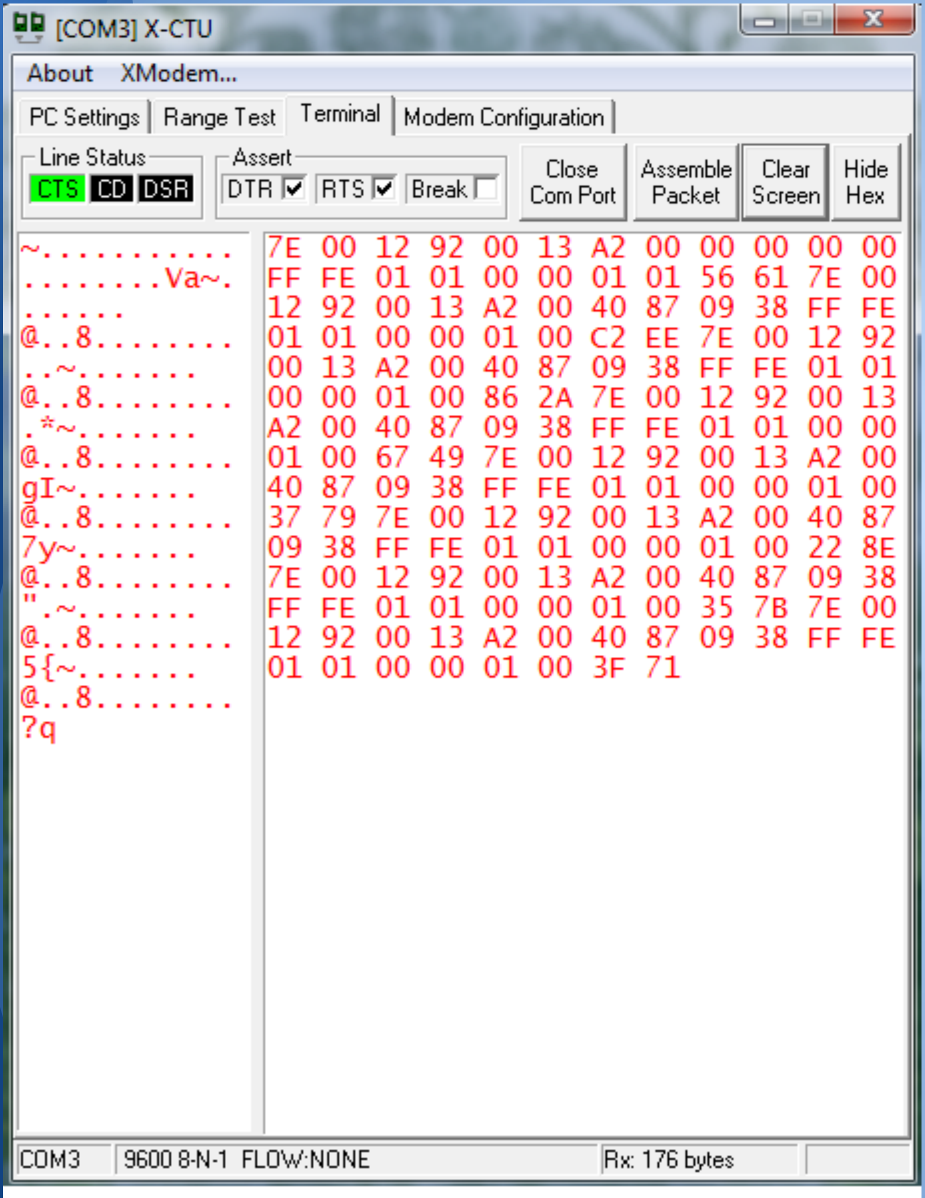
Coordinator



```
+++OK    % Enter Command Mode
ATID8    % Set PAN ID
OK
ATCHB    %Set Channel
OK
ATSM0    %Set Sleep Mode
OK
ATDH13A200  %Set Destination High
OK
ATDL40870936  %Set Destination Low
OK
ATD02    %Set A/D Register to Sample Analog
OK
ATIR64   %Set Sample Rate to every 100ms
OK
ATWR     % Write to Non-Volatile Memory
OK
ATCN     % Exit Command Mode
```

End Device

Xbee *in Action*

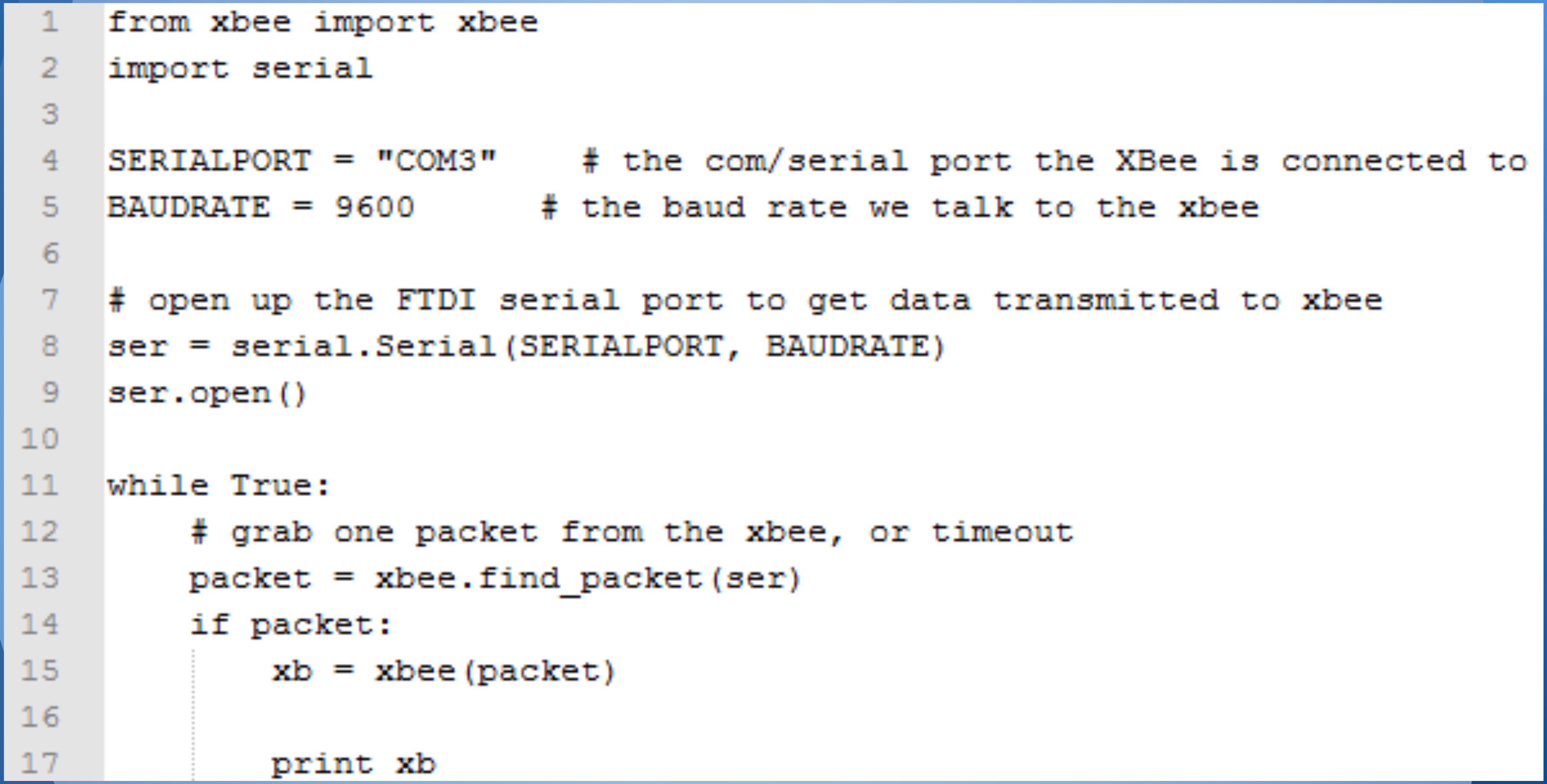


```
~...........     7E 00 12 92 00 13 A2 00 00 00 00 00
........Va~.     FF FE 01 01 00 00 01 01 56 61 7E 00
......           12 92 00 13 A2 00 40 87 09 38 FF FE
@..8........     01 01 00 00 01 00 C2 EE 7E 00 12 92
..~.......       00 13 A2 00 40 87 09 38 FF FE 01 01
@..8........     00 00 01 00 86 2A 7E 00 12 92 00 13
.*~.......       A2 00 40 87 09 38 FF FE 01 01 00 00
@..8........     01 00 67 49 7E 00 12 92 00 13 A2 00
gI~.......       40 87 09 38 FF FE 01 01 00 00 01 00
@..8........     37 79 7E 00 12 92 00 13 A2 00 40 87
7y~.......       09 38 FF FE 01 01 00 00 01 00 22 8E
@..8........     7E 00 12 92 00 13 A2 00 40 87 09 38
".~.......       FF FE 01 01 00 00 01 00 35 7B 7E 00
@..8........     12 92 00 13 A2 00 40 87 09 38 FF FE
5{~.......       01 01 00 00 01 00 3F 71
@..8........
?q
```

COM3 | 9600 8-N-1 FLOW:NONE | Rx: 176 bytes

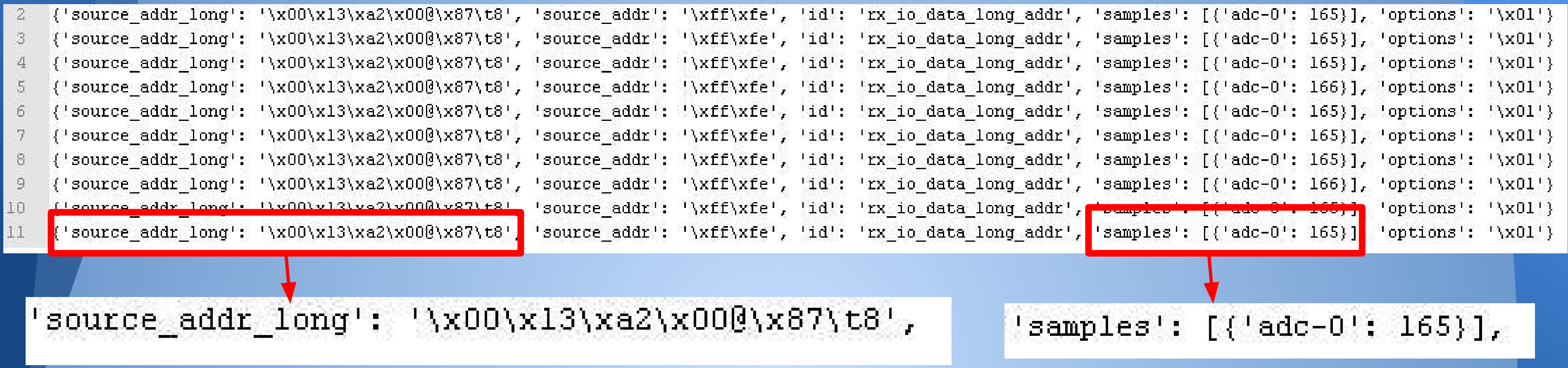
Results of sensing a voltage at the node's ADC pin.

Getting useful data:



```
from xbee import xbee
import serial

SERIALPORT = "COM3"    # the com/serial port the XBee is connected to
BAUDRATE = 9600      # the baud rate we talk to the xbee

# open up the FTDI serial port to get data transmitted to xbee
ser = serial.Serial(SERIALPORT, BAUDRATE)
ser.open()

while True:
    # grab one packet from the xbee, or timeout
    packet = xbee.find_packet(ser)
    if packet:
        xb = xbee(packet)

        print xb
```

Getting useful data:



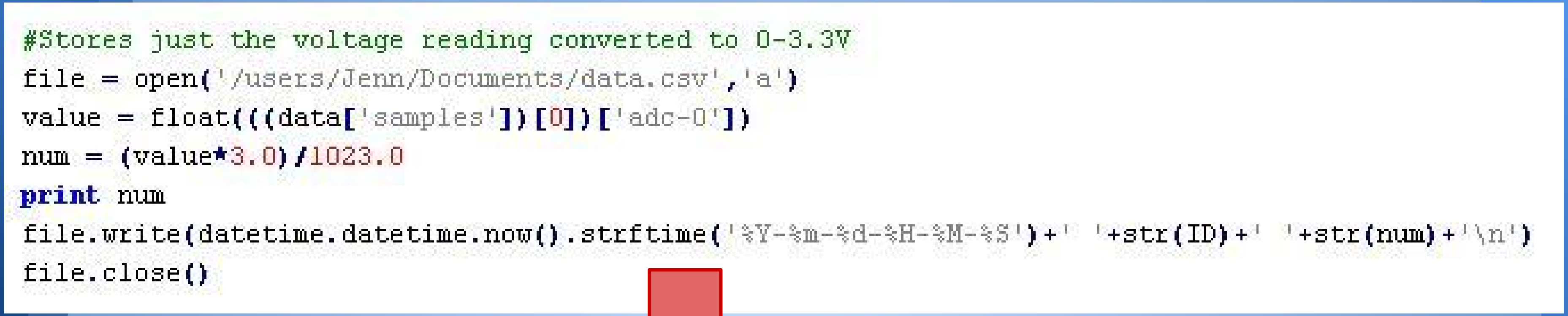
```
{'source_addr_long': '\x00\x13\xa2\x00@\x87\t8', 'source_addr': '\xff\xfe', 'id': 'rx_io_data_long_addr', 'samples': [{'adc-0': 165}], 'options': '\x01'}
{'source_addr_long': '\x00\x13\xa2\x00@\x87\t8', 'source_addr': '\xff\xfe', 'id': 'rx_io_data_long_addr', 'samples': [{'adc-0': 165}], 'options': '\x01'}
{'source_addr_long': '\x00\x13\xa2\x00@\x87\t8', 'source_addr': '\xff\xfe', 'id': 'rx_io_data_long_addr', 'samples': [{'adc-0': 165}], 'options': '\x01'}
{'source_addr_long': '\x00\x13\xa2\x00@\x87\t8', 'source_addr': '\xff\xfe', 'id': 'rx_io_data_long_addr', 'samples': [{'adc-0': 166}], 'options': '\x01'}
{'source_addr_long': '\x00\x13\xa2\x00@\x87\t8', 'source_addr': '\xff\xfe', 'id': 'rx_io_data_long_addr', 'samples': [{'adc-0': 165}], 'options': '\x01'}
{'source_addr_long': '\x00\x13\xa2\x00@\x87\t8', 'source_addr': '\xff\xfe', 'id': 'rx_io_data_long_addr', 'samples': [{'adc-0': 165}], 'options': '\x01'}
{'source_addr_long': '\x00\x13\xa2\x00@\x87\t8', 'source_addr': '\xff\xfe', 'id': 'rx_io_data_long_addr', 'samples': [{'adc-0': 165}], 'options': '\x01'}
{'source_addr_long': '\x00\x13\xa2\x00@\x87\t8', 'source_addr': '\xff\xfe', 'id': 'rx_io_data_long_addr', 'samples': [{'adc-0': 166}], 'options': '\x01'}
{'source_addr_long': '\x00\x13\xa2\x00@\x87\t8', 'source_addr': '\xff\xfe', 'id': 'rx_io_data_long_addr', 'samples': [{'adc-0': 165}], 'options': '\x01'}
{'source_addr_long': '\x00\x13\xa2\x00@\x87\t8', 'source_addr': '\xff\xfe', 'id': 'rx_io_data_long_addr', 'samples': [{'adc-0': 165}], 'options': '\x01'}
```

```
'source_addr_long': '\x00\x13\xa2\x00@\x87\t8',
```

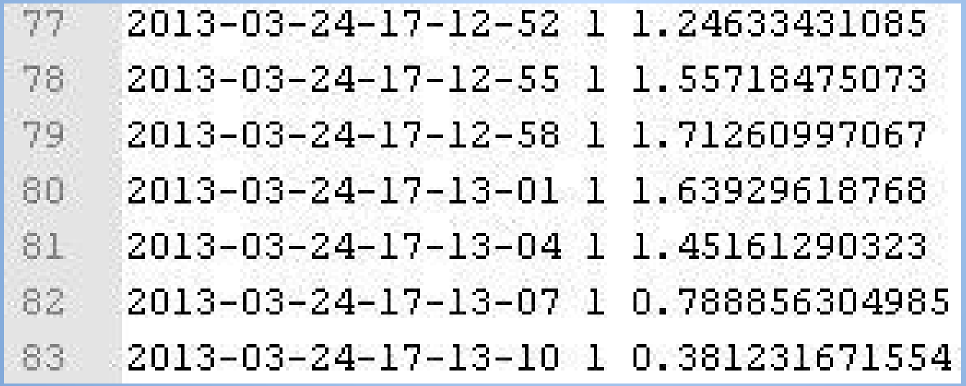
```
'samples': [{'adc-0': 165}],
```

- Parse packet to get only source address and sample data
- Add date/time stamp
- Store everything in a file

Parsing and Storing:



```
#Stores just the voltage reading converted to 0-3.3V
file = open('/users/Jenn/Documents/data.csv','a')
value = float(((data['samples'])[0])['adc-0'])
num = (value*3.0)/1023.0
print num
file.write(datetime.datetime.now().strftime('%Y-%m-%d-%H-%M-%S')+' '+str(ID)+' '+str(num)+'\n')
file.close()
```



```
2013-03-24-17-12-52 1 1.24633431085
2013-03-24-17-12-55 1 1.55718475073
2013-03-24-17-12-58 1 1.71260997067
2013-03-24-17-13-01 1 1.63929618768
2013-03-24-17-13-04 1 1.45161290323
2013-03-24-17-13-07 1 0.788856304985
2013-03-24-17-13-10 1 0.381231671554
```

Year-Month-Day-Hour-Minute-Second Node ID Voltage

Summary

- Pick Xbee for your networking needs
- Connect to power and your sensor
- Configure the Coordinator and Remote Nodes
- Use Python script to see data

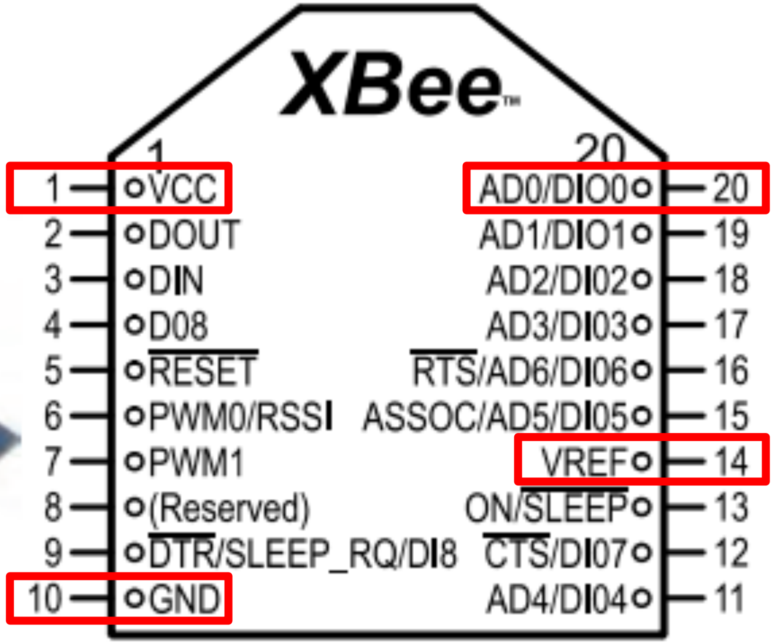
Questions?

Appendix

References

- Xbee DigiMesh 2.4 RF Module Datasheet [Link]
- Xbee Family Features Comparison [Link]
- Using XBee Radios for Wireless Acceleration Measurements [Link]
- Tweet-A-Watt [Link]

Hardware Setup



Configuration

(AT)	What it is:	Options	Example
ID	PAN ID	0-0x7FFF	8
CH	Channel	0x0B-0x1A	B
SM	Sleep Mode	0-Normal (no sleep) 7-Sleep Support Node 8-Cyclic Sleep	0

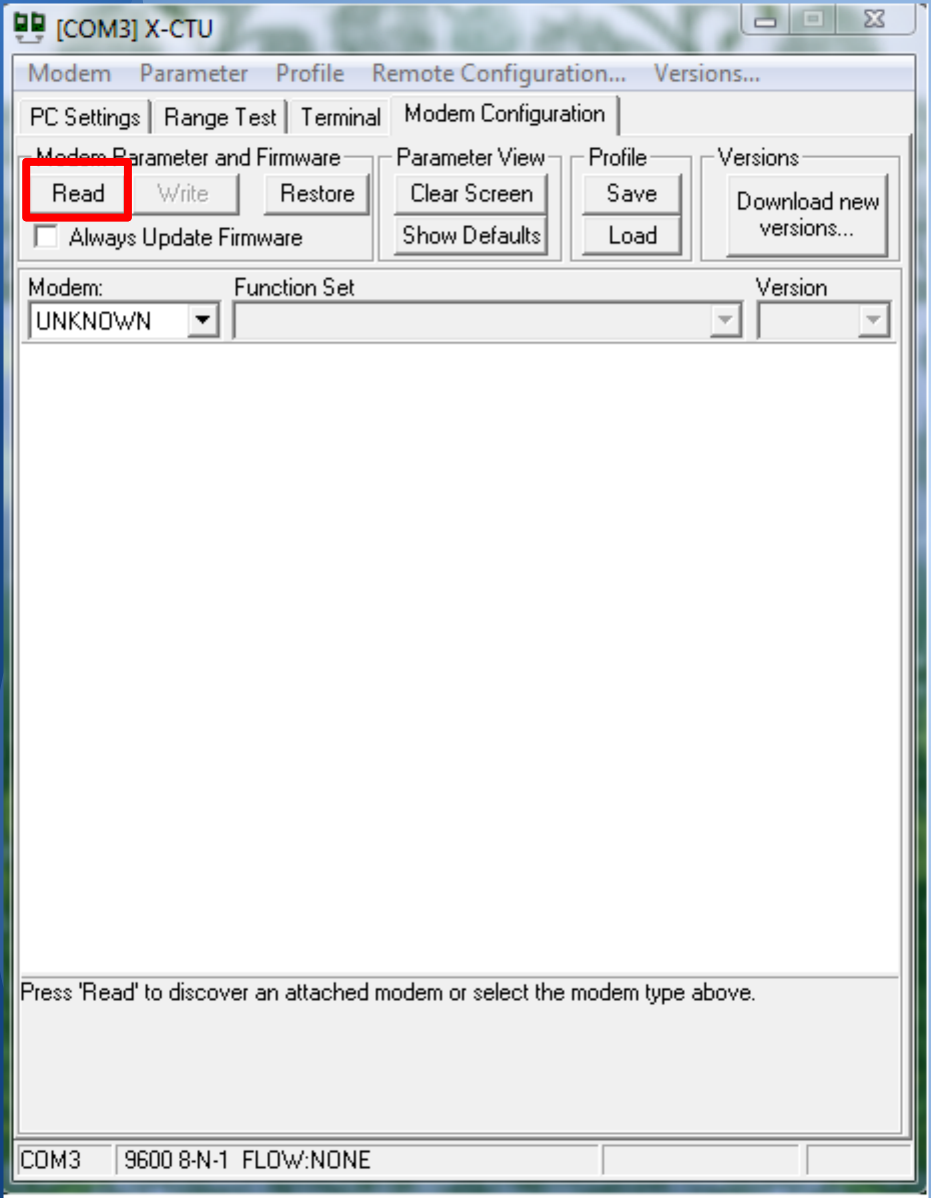
Configuration

(AT)	What it is:	Options	Example
DH	Destination Address High	0-0xFFFFFFFF	0013A200
DL	Destination Address Low	0-0xFFFFFFFF	40870936
IR	I/O Sampling Rate	0-0xFFFF (ms)	64 (100ms)

Configuration

(AT)	What it is:	Options	Example
DO	AD0/ DIO0	0-Disabled 1-Commissioning Button Enable 2-Analog input 3-Digital Input 4-Digital Output low 5-Digital Output high	2
AP	API Mode	0-Off 1-On 2-On with escaped sequences	1

Configuration



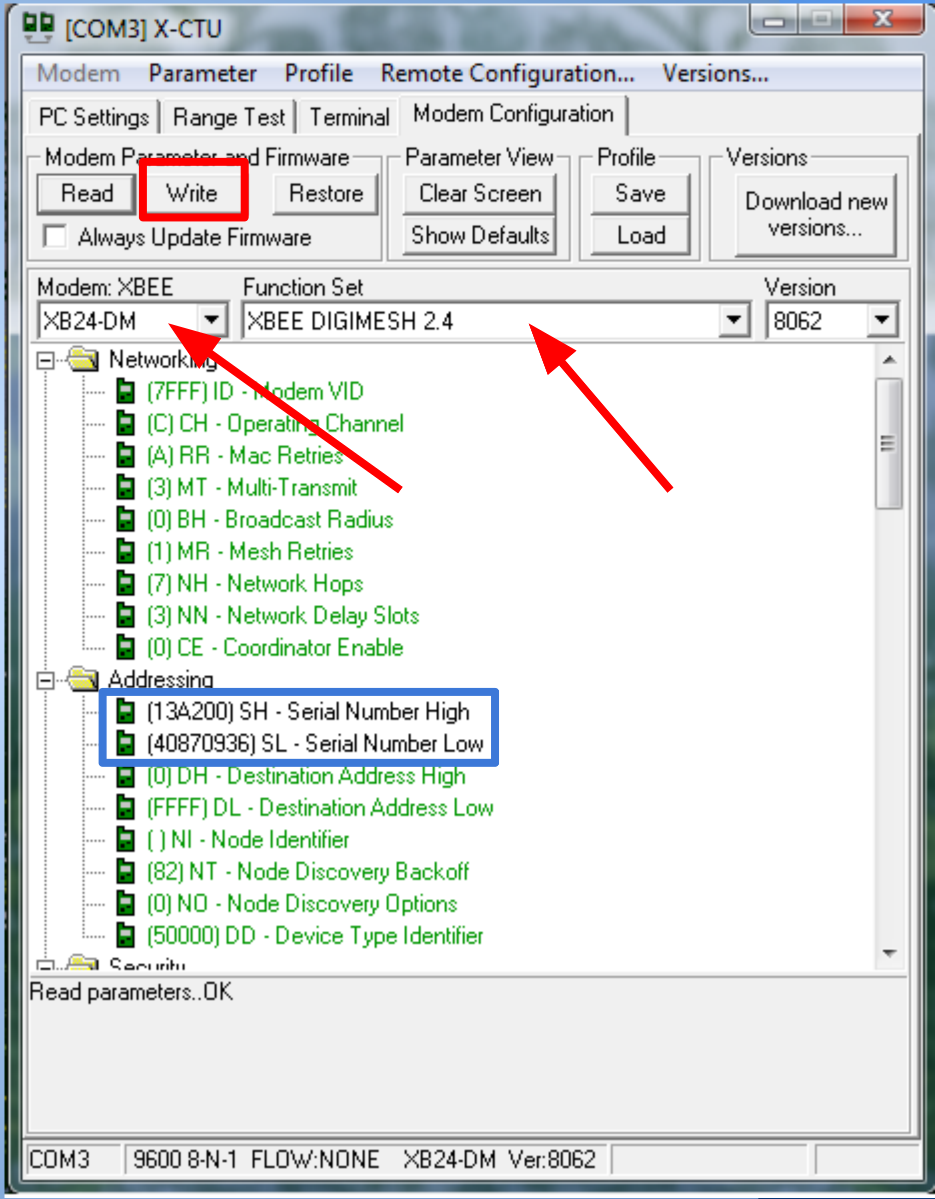
Modem: UNKNOWN

Function Set

Version

Press 'Read' to discover an attached modem or select the modem type above.

```
COM3   9600 8-N-1  FLOW:NONE
```



Modem: XBEE XB24-DM

Function Set XBEE DIGIMESH 2.4

Version 8062

- Networking
  - (7FFF) ID - Modem VID
  - (C) CH - Operating Channel
  - (A) RR - Mac Retries
  - (3) MT - Multi-Transmit
  - (0) BH - Broadcast Radius
  - (1) MR - Mesh Retries
  - (7) NH - Network Hops
  - (3) NN - Network Delay Slots
  - (0) CE - Coordinator Enable
- Addressing
  - (13A200) SH - Serial Number High
  - (40870936) SL - Serial Number Low
  - (0) DH - Destination Address High
  - (FFFF) DL - Destination Address Low
  - ( ) NI - Node Identifier
  - (82) NT - Node Discovery Backoff
  - (0) NO - Node Discovery Options
  - (50000) DD - Device Type Identifier
- Security

Read parameters..OK

```
COM3   9600 8-N-1  FLOW:NONE   XB24-DM  Ver:8062
```