XBee Basics

Rob Faludi

Moving Data by Radio

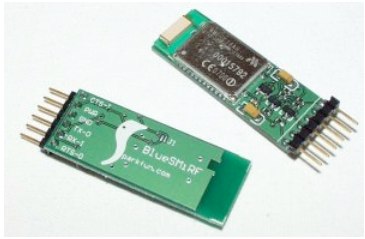
toys	wearables	performance
portables	emergent systems	anything spinning
network objects	sensors	audio/video
feedback	remotes	context awareness

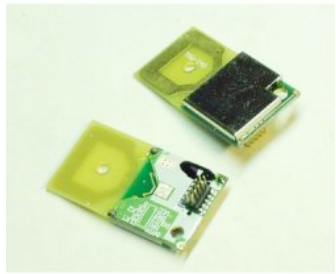
What Do We Want?

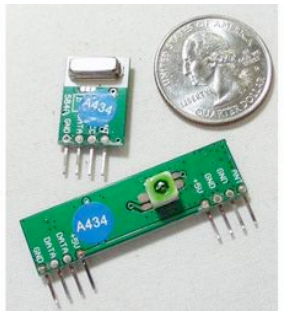
wireless	easy communication	reliability
low power	addressing	broadcast
small	standardized	cheap
bandwidth	fast	routing

Existing Methods for Device Communication

- Bluetooth
- "RF"
- XPort TCP/IP
- MatchPort TCP/IP
- Cell Phone Data GPRS







ZigBee & 802.15.4

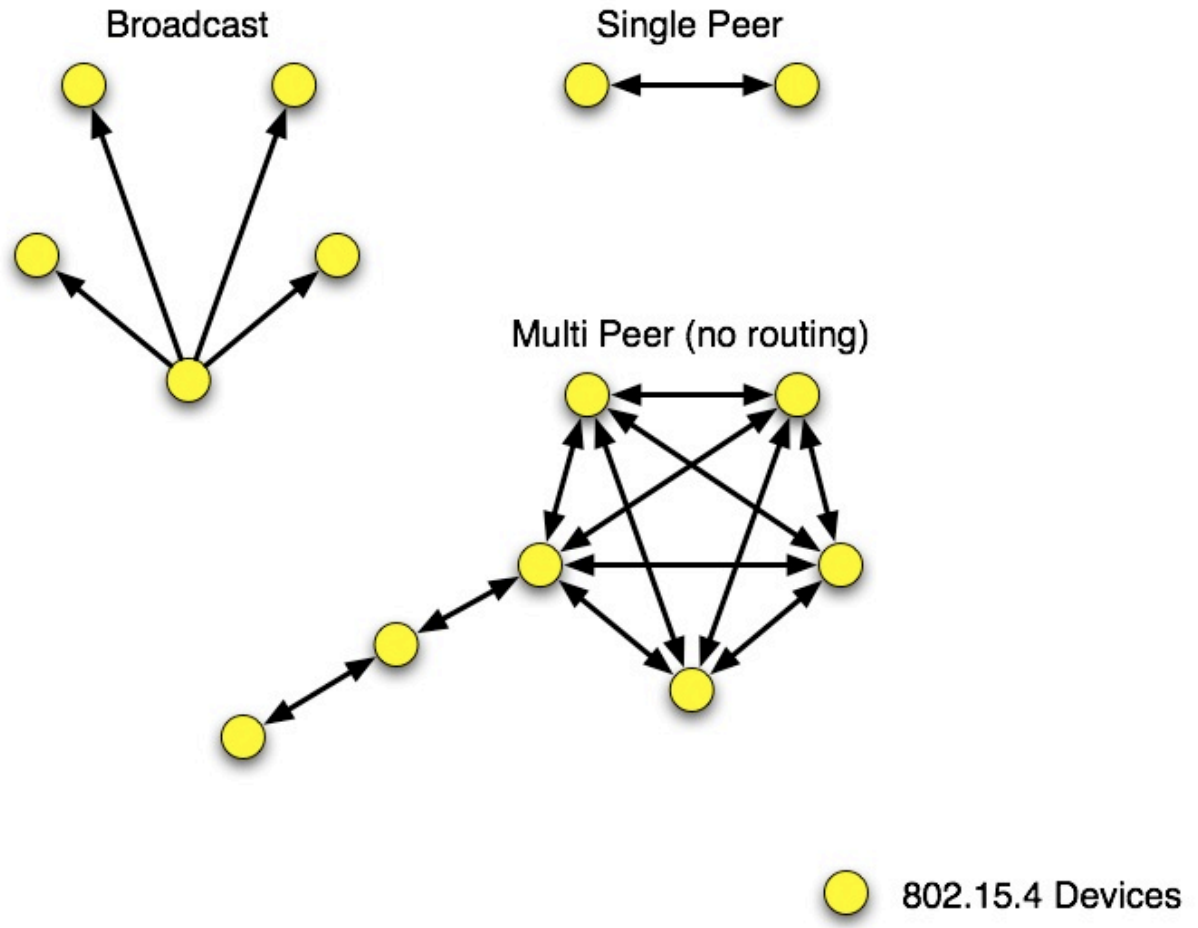
- ZigBee is built on top of the IEEE 802.15.4 protocol
- XBee radios are available with or without ZigBee
- XBee 802.15.4 vs. ZNet 2.5
- Both ways are useful

802.15.4

- low power
- addressing
- cheap
- wireless
- small
- standardized

802.15.4 Topologies

- single peer
- multi-peer
- broadcast



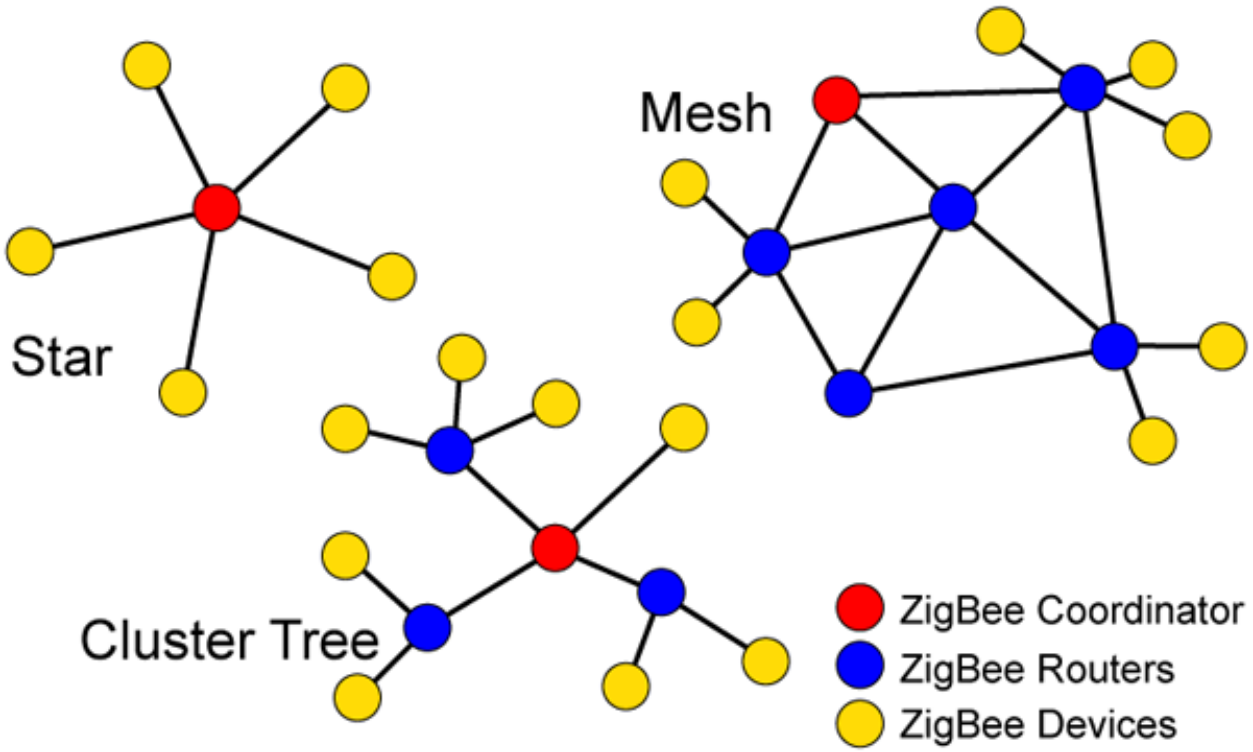
ZigBee

- routing
- self-healing mesh
- ad-hoc network creation

ZigBee™ Alliance

ZigBee Topologies

- peer
- star
- mesh
- routing

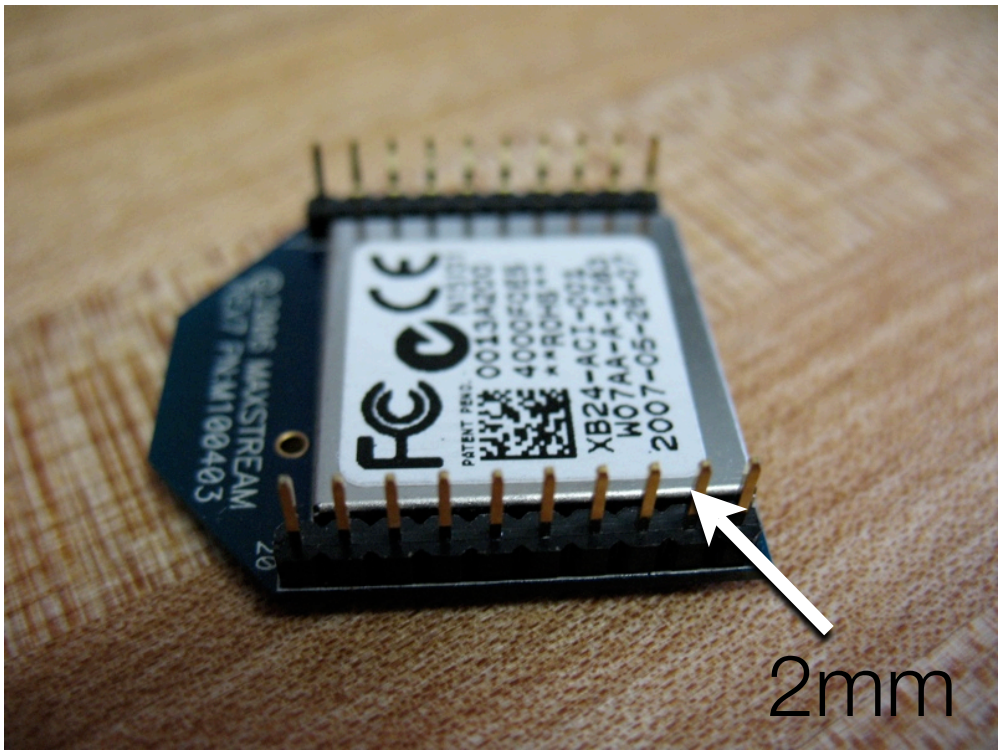


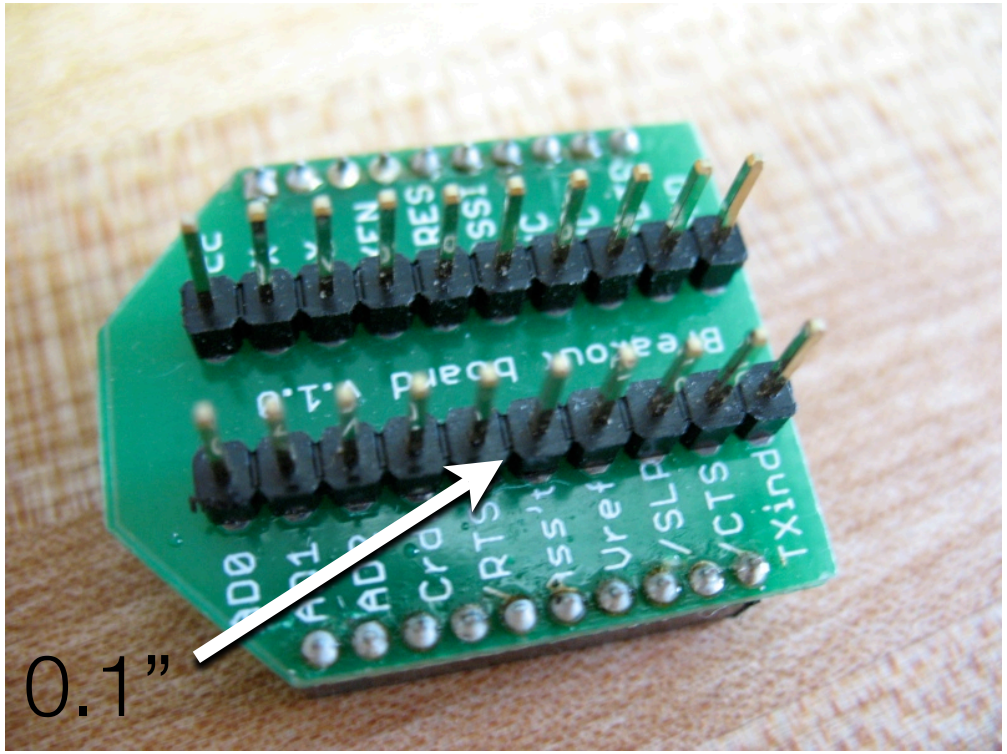
How Do I Make One?

Materials

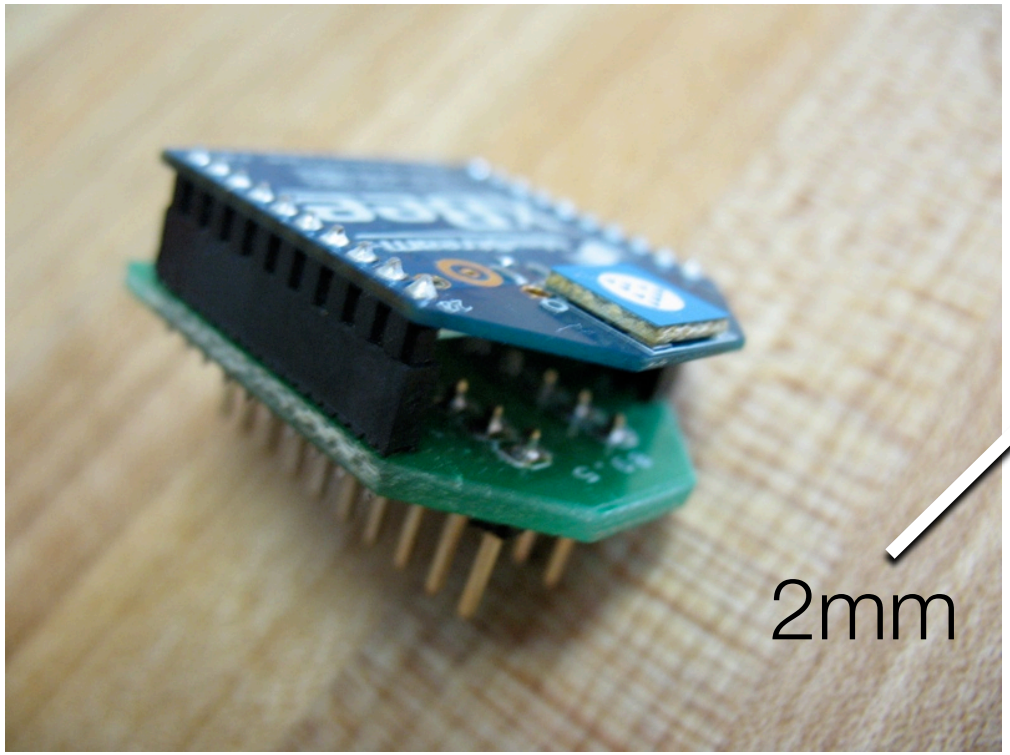
- XBee OEM Module (30-100 m range) $19
  XBee Pro (100m - 1.6 km range) $32
    - Digi: http://www.digi.com
- Breakout Board, 2mm to 10 mil pin spacing. From me or Spark Fun
- Female headers 2mm from me or Spark Fun
- Male headers 10 mil (in stock at ITP)

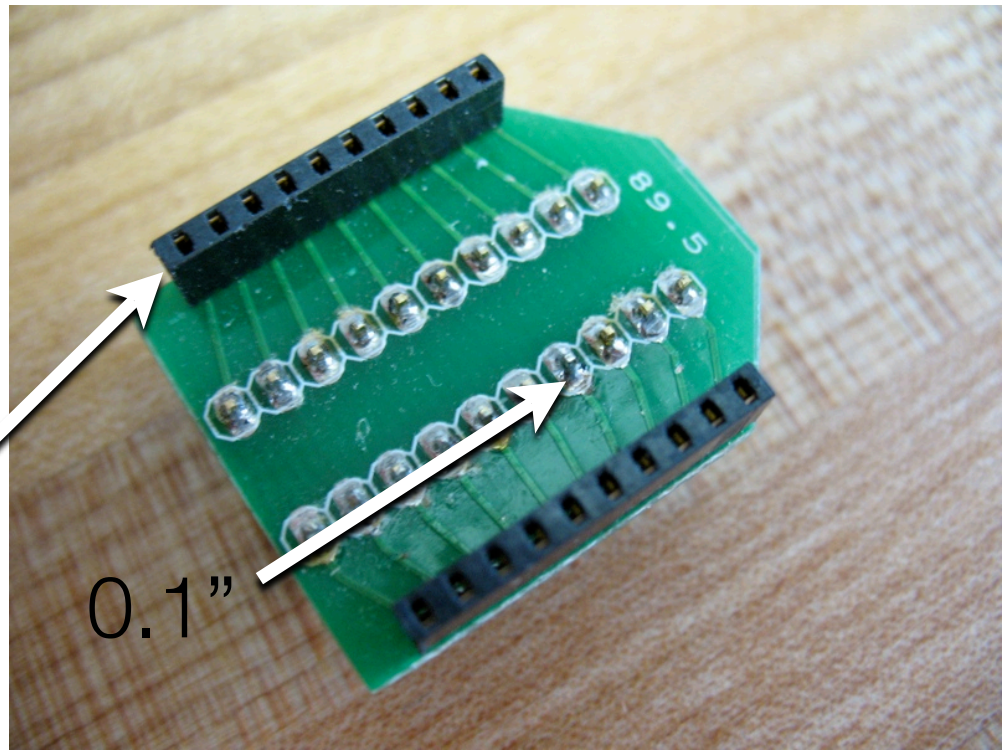
Soldering Breakout Boards: pin spacing



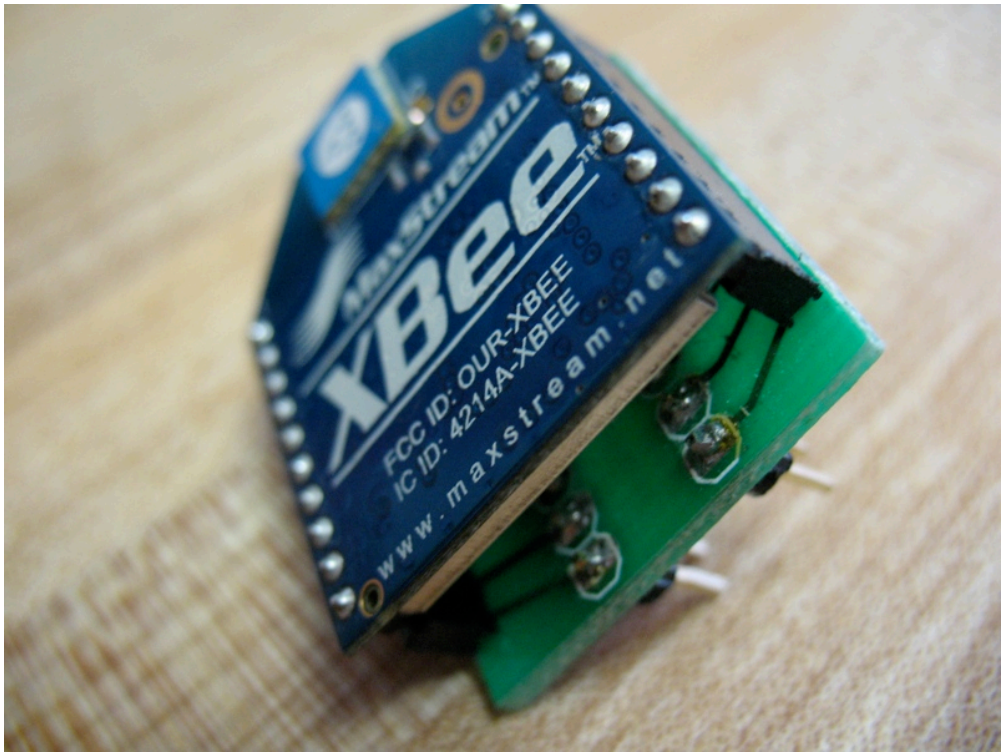


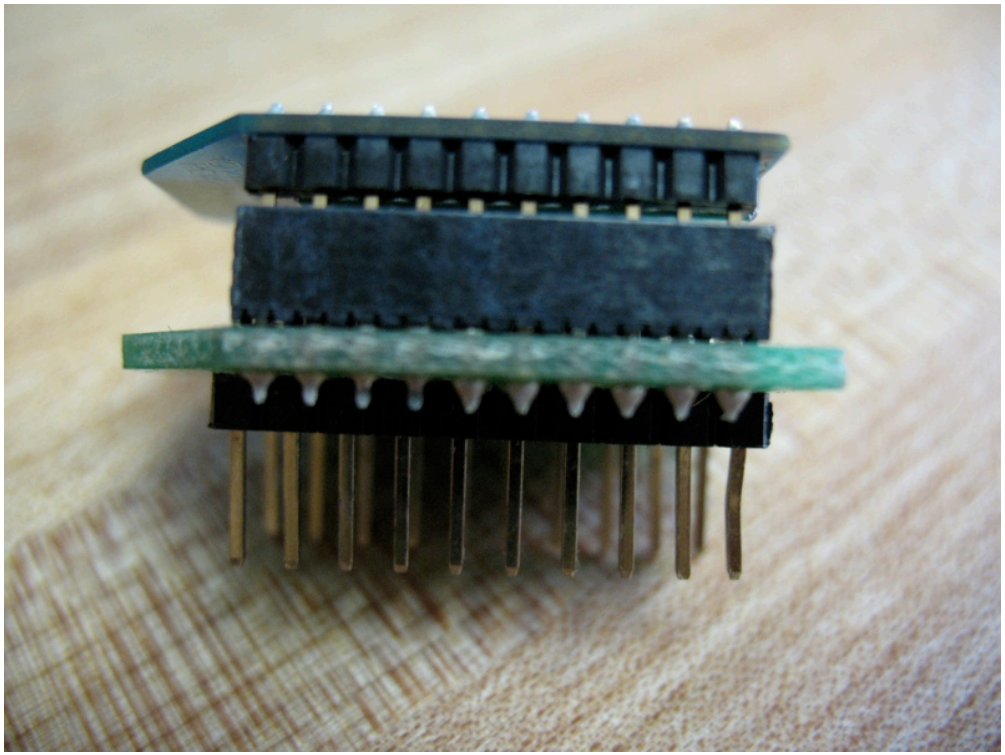
Soldering Breakout Boards: headers



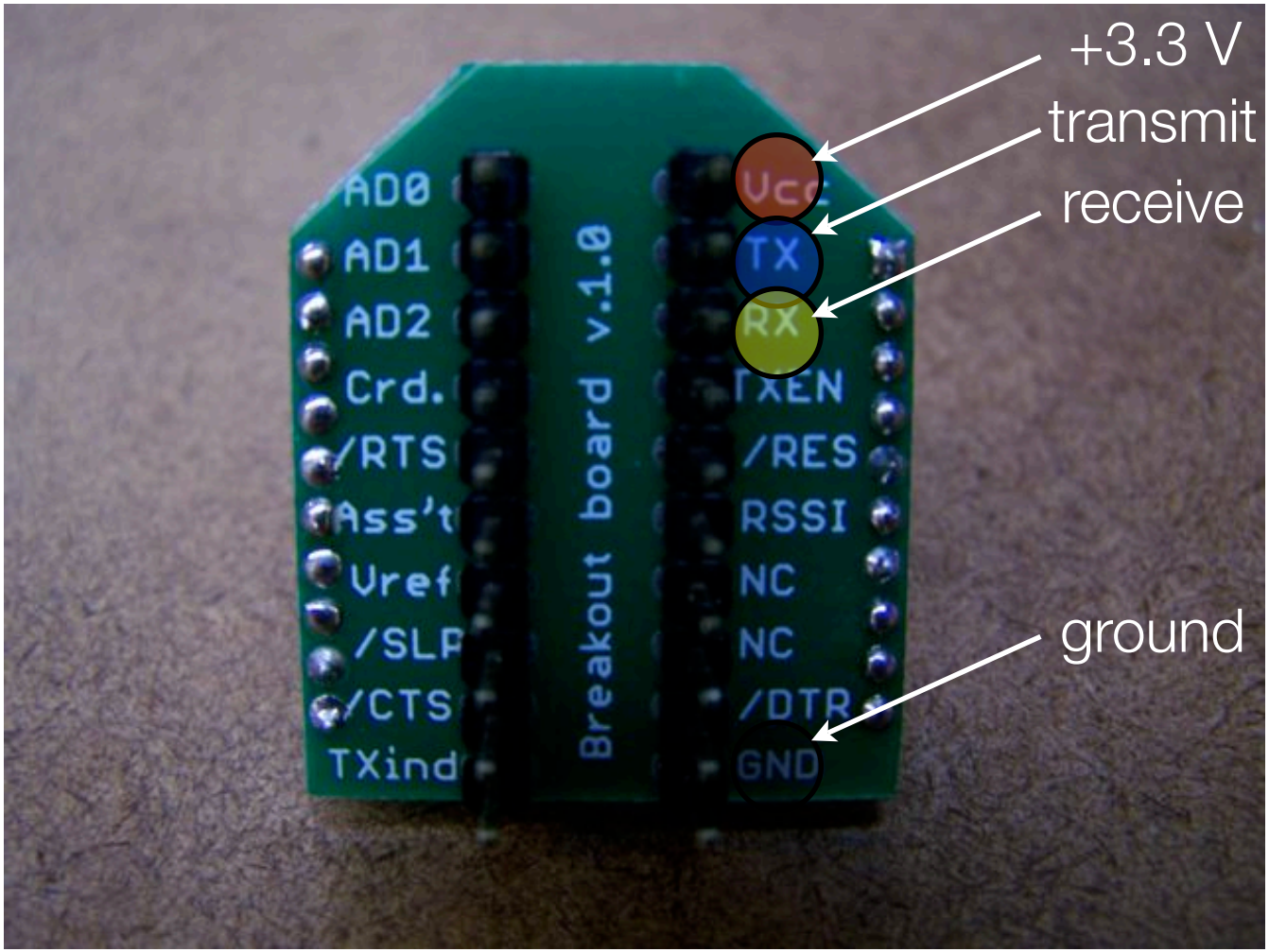


Soldering Breakout Boards: finished





Wiring


+3.3 V
transmit
receive
ground
AD0
AD1
AD2
Crd.
/RTS
Ass't
Vref
/SLP
/CTS
TXind
Breakout board v.1.0
Vcc
TX
RX
TXEN
/RES
RSSI
NC
NC
/DTR
GND

Remember!

- Use only +3.3 Volts. The regulator often has a different pin arrangement: G-O-I
- Always use decoupling capacitors. The radios often don't work without them.
- You can't send infinitely fast. Try putting a 10 ms delay into your loop.
- XBee TX goes to Arduino RX and vice versa.
- Arduino can run on 3.3 Volts (use a mini or breadboard with NG bootloader)

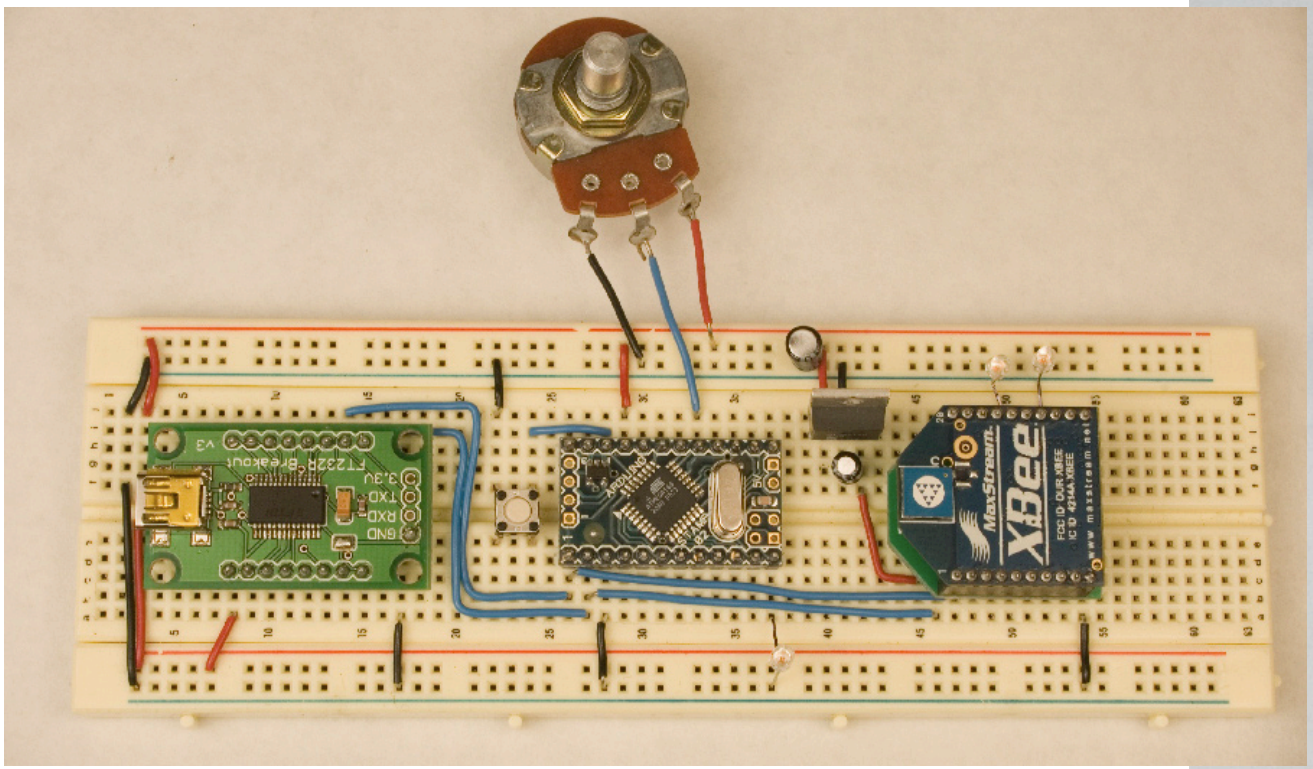
Instructions

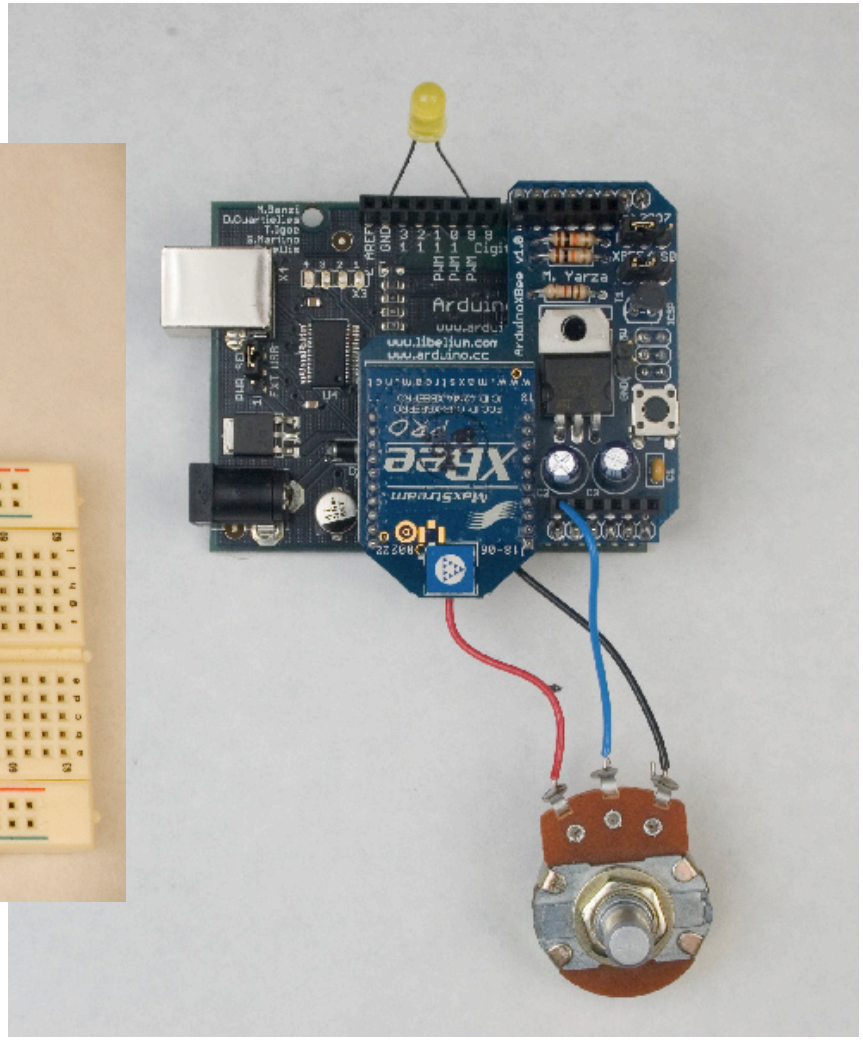
- XBee Practical Example: Paired communication between two microcontrollers. Includes building, wiring and code for PIC and Arduino

- Making Things Talk by Tom Igoe

- I/O Example on my blog, or in the XBee manual section 2.2

XBee Send/Receive





Serial Terminal Programs

- Processing: http://rob.faludi.com/teaching/cmn/code/XBee_Terminal.pde
- Z-Term: http://homepage.mac.com/dalverson/zterm/
- HyperTerm: Windows Start Menu, Accessories, Communication
- screen: Terminal program on the Mac (or Linux)
- X-CTU: http://www.digi.com/support/productdetl.jsp?pid=3352&osvid=57&tp=4&s=316
- plenty of others

Baud, Bits and Parity

- Setting different baud rates: 9600
- Stop bits: 1
- Parity: None
- Flow control: none for now...

Data Mode vs. Command Mode

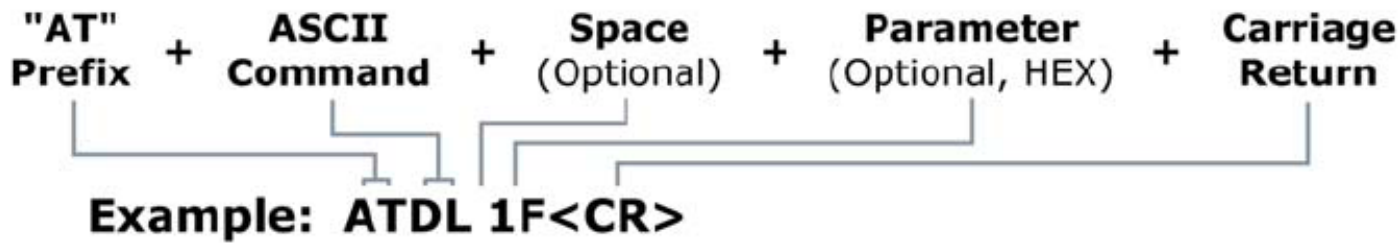
- Idle Mode, transmit and receive data
- Command Mode, talk to the XBee itself
  - +++ "*Yo, XBee*"
  - AT "*Attention!*" (Hayes command set)

Some AT Commands

- AT -> OK
- ATMY -> my address
- ATDH, ATDL -> destination address hi/lo
- ATID -> personal area network ID
- ATCN -> end command mode

AT Command Format

Figure 2-08. Syntax for sending AT Commands



Method 1 (One line per command)

Send AT Command	System Response
+++	OK <CR> (Enter into Command Mode)
ATDL <Enter>	{current value} <CR> (Read Destination Address Low)
ATDL1A0D <Enter>	OK <CR> (Modify Destination Address Low)
ATWR <Enter>	OK <CR> (Write to non-volatile memory)
ATCN <Enter>	OK <CR> (Exit Command Mode)

Method 2 (Multiple commands on one line)

Send AT Command	System Response
+++	OK <CR> (Enter into Command Mode)
ATDL <Enter>	{current value} <CR> (Read Destination Address Low)
ATDL1A0D,WR,CN <Enter>	OK, OK, OK <CR> (Command execution is triggered upon each instance of the comma)

Hexadecimals

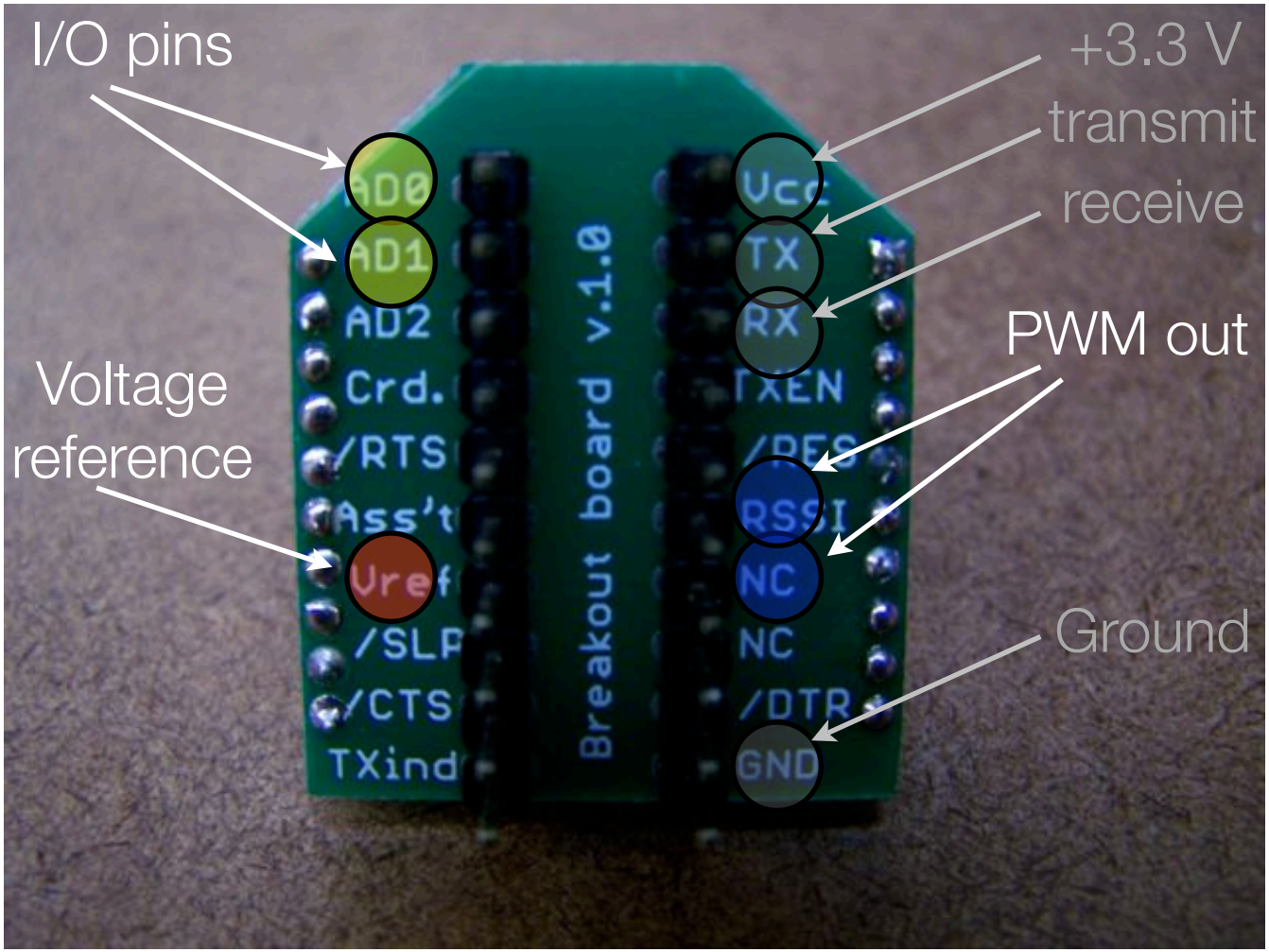
- Just like decimals, but count from 0 to 15 in each position
- Since there's no existing single numeral representing 10 - 15, use A - F instead
- A = 10, B=11, C=12 ... F=15
- A1 = 161, common notation: 0xA1
- What does BFF equal? What does it look like?
- Calculators on Mac & Windows

Example:
Remote Rotation

I/O Why

- Why:
  - Save space, save power, save weight and save money
  - Reduce complications
- Why not:
  - Limited inputs/outputs
  - No access to logic
  - Each radio must be manually configured

Input/Output Wiring



I/O AT Commands

- ATD0...D8 -> configure pins for I/O
- ATIR -> sample rate
- ATIT -> samples before transmit
- ATP0...P1 -> PWM configuration
- ATIU -> I/O output enable (UART)
- ATIA -> I/O input address

Example Configuration

- ATID3456 (PAN ID)
  ATMY1 my address 1
  ATDL2 destination address 2
  ATD02 output 0 in analog mode
  ATD13 output 1 in digital out mode
  ATIR14 sample rate 20 milliseconds (hex 14)
  ATIT5 samples before transmit 5

- ATID3456 (PAN ID)
  ATMY2 my address 2
  ATDL1 destination address 1
  ATP02 PWM 0 in PWM mode
  ATD15 output 1 in digital out high mode
  ATIU1 I/O output enabled
  ATIA1 I/O input from address 1

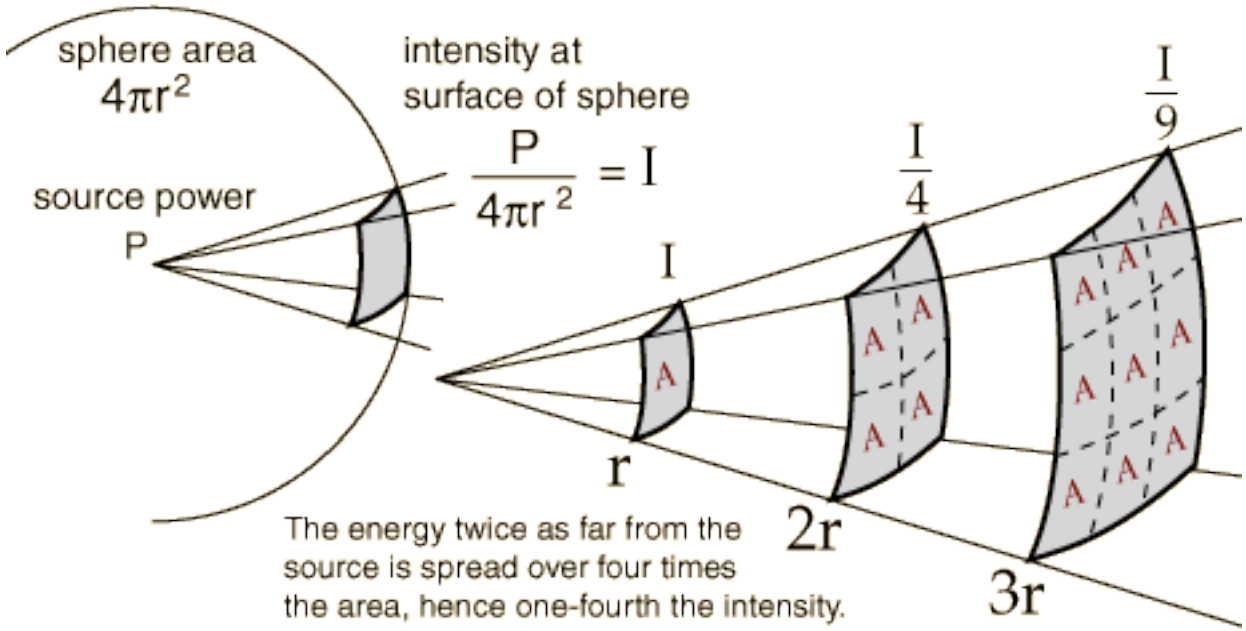
Radio Communications

- What is radio?
    - electromagnetic waves
    - no medium required
- Modulation
- Well-described mystery: “air waves” “wireless” “ethereal communication”
- posters

Why Wireless?

- why wireless (mesh ≠ wireless)

- inverse square law



- what technologies can be used for device communication?

API Mode

- Powerful, steeper learning curve
- Data wrapped together with commands, addressing and status information

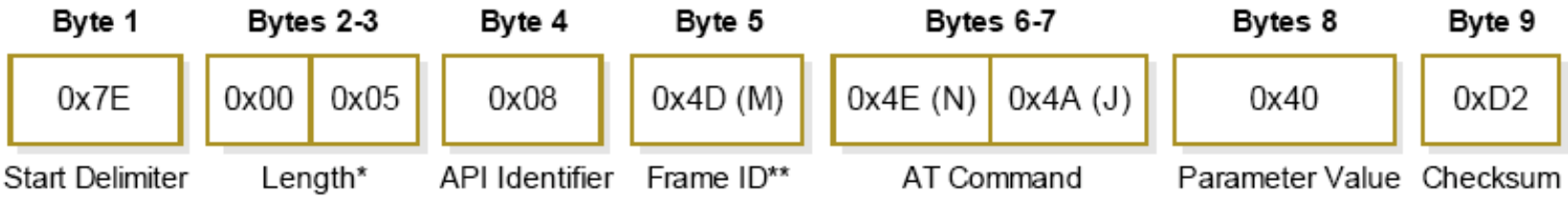
API Mode Format

Figure 4-01. UART Data Frame Structure:

MSB = Most Significant Byte, LSB = Least Significant Byte

Any data received prior to the start delimiter is silently discarded. If the frame is not received correctly or if the checksum fails, the module will reply with a module status frame indicating the nature of the failure.

Figure 4-07. Example: API frames when modifying the NJ parameter value of the module.



** Length [Bytes] = API Identifier + Frame ID + AT Command + Parameter Value*

*** "M" value was arbitrarily selected.*

*ATNJ = node join

Protocols

- Sending
- Flow control
- Call / response
- Broadcast
- Start / stop
- Checksums
- Collisions