Sensitive Buildings

Instructor: Rob Faludi

Plan for Today

- Doorbell Projects: review
- I/O Mode
- I/O Demo
- Voltage Dividers
- API Mode Overview
- API Mode Details
- Readings & Assignments

Doorbell Projects Review

I/O Mode

Direct, Indirect, Subtext

- What data can we sense directly?
- How about inferences that we can make from the data?
- What's the subtext of the data? What can we infer from the inference?

I/O Intro: ZigBee

- For simple input and/or output
- Ten digital input/outputs
- Four analog inputs
- No analog outputs on ZigBee
- But not all at once! Pins are shared.

I/O Why

- Why:
 - Save space, save power, save weight and save money
 - Reduce complications
- Why not:
 - Limited inputs/outputs
 - No access to logic
 - No analog output on ZigBee radios

Input/Output Wiring: ZigBee



Settting I/O Pins

- ATDx 0 Disabled
- ATDx 1 Built-in Function (sometimes)
- ATDx 2 Analog Input (sometimes)
- ATDx 3 Digital Input
- ATDx 4 Digital <u>Output</u>, low to start with
- ATDx 5 Digital <u>Output</u>, high to start with
 - ...so ATD32 would set which pin to which mode?

I/O AT Commands: ZigBee

- ATD0...D7 -> configure pins for I/O (D8 and D9 not supported yet)
- ATP0...P1 -> configure pins 10 11 for I/O (P3 not supported yet)
- ATIR -> sample rate
- samples before transmit is always 1
- destination address receives sample info
- ALL PINS READ BETWEEN 0 AND 1.2 VOLTS ONLY

Example Configuration

- <u>SENDER</u>: ATID3456 (PAN ID)
 ATDH -> set to SH of partner radio
 ATDL -> set to SL of partner radio
 ATJV1 -> rejoin with coordinator on startup
 ATD02 pin 0 in analog in mode
 ATD13 pin 1 in digital in mode
 ATIR64 sample rate 100 millisecs (hex 64)
- <u>RECEIVER</u>

ATID3456 (PAN ID) ATDH -> set to SH of partner radio ATDL -> set to SL of partner radio

I/O Demo

XBee ZigBees inputs are 1.2V range

Voltage Divider to map 3.3V range to 1.2V range



Romantic Lighting Sensor



API Mode Overview

API Mode

- Application Programming Interface
 - "An application programming interface (API) is a source code interface that an operating system or library provides to support requests for services to be made of it by computer programs."

http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=43487

- XBees in API mode are ready to talk to computers and microcontrollers
 - structured
 - predictable
 - reliable



API Structure

- Used in serial communications with the XBee radio
- Frames of data
 - envelope structure contains data with metadata inside a constrained format
- Radio must be in API Mode
 - AT command ATAP 1 on Series 1 radios
 - API firmware on Series 2 radios

Why API

• Rather than:

```
delay(1100);
// put the XBee in command mode
Serial.print("+++");
delay(1100);
if (checkFor("OK", 1000)) {
   Serial.println("ATID7777,CN");
   if (checkFor("OK", 1000)) {
     // if an OK was received then continue
     debugPrintln("SetupOK");
     success = true;
   }
}
```

• With a library you just write:

```
sendCommand(ID,0x7777);
```

API & AT can Happily Co-exist



API Mode Details

Protocols

- Sending
- Start
- Length
- Contents
- Checksums
- Collisions

Envelope Has:

• From address, to address, outside, inside, size, contents, error check

May E. Hill 1205 Jindew St, N.E., CHINGS Washington, D.C. (SHING) 3 1919 5.120.57 Lieut. Daniel Grafton Hill Jr. Co. M. 368th Infantry american Expeditionary Frees via New York.

API Basic Frame Envelope



Start Byte

- 0x7E --> also known as the tilde in ASCII: ~
- First thing to do is look for it:

```
// ARDUINO VERSION:
if (Serial.available() > 0) { // if a byte is waiting in the buffer
    inByte = Serial.read(); // read a byte from the buffer
    if (inByte == 0x7E) {
       // we're at the start of an API frame!
       // add more code here
    }
  }
  // PROCESSING VERSION:
if (port.available() > 0 {
  int inByte = port.read();
    if (inByte == 0x7E) {
       // we're at the start of an API frame!
       // add more code here
}
```

Length Bytes

- MSB: the Most Significant Byte
 - the big part of the number
- LSB: the Least Significant Byte
 - the small part of the number
- bit shift MSB to the right and add it to LSB

```
// PROCESSING VERSION:
int lengthMSB = port.read(); // high byte for length of packet
int lengthLSB = port.read(); // low byte for length of packet
```

```
int lengthTotal = (lengthMSB * 0x100) + lengthLSB; //multiply and add for total
```

API Identifier

- Specifies the remaining structure of the frame
 - modem status: 0x8A
 - AT command (immediate): 0x08
 - AT command (queued): 0x09
 - AT command response: 0x88
 - TX request: 0x10
 - TX status response: 0x8B
 - RX packet: 0x90
 - RX packet I/O data: 0x92

```
// PROCESSING VERSION:
int API_ID = port.read(); // API Identifier indicates type of packet received
```

Identifier-specific Data

- Structures are different for each API identifier and might include:
 - addressing information (333B)
 - status information (received OK)
 - source information (broadcast packet)
 - unstructured data ("Hello World, this is Rob!")
 - structured data (typically for I/O packets)

Checksum

- Simple check to detect errors
- To calculate: Not including frame delimiters and length, add all bytes keeping only the lowest 8 bits of the result and subtract from 0xFF.
- To verify: Add all bytes (include checksum, but not the delimiter and length). If the checksum is correct, the sum will equal 0xFF.

```
// PROCESSING VERSION:
int localChecksum = (API_ID + addrMSB + addrLSB + RSSI + options + dataSum);
int checksum = port.read();
localChecksum = byte(0xFF -localChecksum);
if ( (byte) checksum - localChecksum == 0) {
  returnVal = dataADC[0];
}
else {
  print("\n\nchecksum error! " + "\n\n");
}
```

Many Kinds of Envelopes



Modem Status: ZigBee



AT Command





AT Response

• Frame ID for the response is the same as the matching AT Command request



More API

TX (Transmit) Request

• Remember that this is a request. Results can be checked by Frame ID



TX Status (Results)

- See if your message was transmitted or not
- Use your Frame ID to see which message is being described

Start Delimiter Lo	angth Frame Data	Checksum			
0x7E MSB	LSB API-specific Structure	1 Byte			
	API identifier identifier-specific i 0x8B cmdData	Data			
Frame ID (Byte 5)	Remote Network Address (Bytes 6-7)	Transmit Retry Count (Byte 8)	Delivery Status (Byte 9)	Discovery Status (Byte 10)	
Identifies UART data frame being reported.	16-bit Network Address the packet was delivered to (if success). If not success, this address matches the Destination Network Address that was provided in the Transmit Request Frame.	The number of application transmission retries that took place.	0x00 = Success 0x02 = CCA Failure 0x15 = Invalid destination endpoint 0x21 = Network ACK Failure 0x22 = Not Joined to Network 0x23 = Self-addressed 0x24 = Address Not Found 0x25 = Route Not Found	0x00 = No Discovery Overhead 0x01 = Address Discovery 0x02 = Route Discovery 0x03 = Address and Route Discovery	

RX Packet

- Maximum of 72 bytes of data per packet
- RF Data section is basis for I/O packets



I/O RX Packet



I/O Digital Channel Mask and Digital Data

Digital Channel Mask (bytes 17-18)*

Bitmask field that indicates which digital IO lines on the remote have sampling enabled (if any)

N/A	N/A	N/A	CD/DIO	PWM/DI	RSSI/DI	N/A	N/A
			12	O11	O10		
CTS/DI	RTS/DI	ASSOC/	DIO4	AD3/DI	AD2/DI	AD1/DI	AD0/DI
07	O6	DIO5		O3	O2	01	O 0
							1 M M

Digital Samples (bytes 20-21, if included)

If the sample set includes any digital IO lines (Digital Channel Mask > 0), these two bytes contain samples for all enabled digital inputs. DIO lines that do not have sampling enabled return 0. Bits in these 2 bytes map the same as they do in the Digital Channels Mask field.

I/O Analog Channel Mask and Analog Samples

Analog Channel Mask

(byte 19)**

Bitmask field that indicates which digital IO lines on the remote have sampling enabled (if any).

	Supply Voltage	N/A	N/A	N/A	AD3	AD2	AD1	AD0	
--	-------------------	-----	-----	-----	-----	-----	-----	-----	--

Analog Samples (2 bytes each sample)

If the sample set includes any analog input lines (Analog Channel Mask > 0), each enabled analog input returns a 2-byte value indicating the A/D measurement of that input. Analog samples are ordered sequentially from AD0/DIO0 to AD3/DIO3, to the supply voltage.

I/O Structure Reviewed

- Num Samples (1 byte)
- Digital Channel Mask (2 bytes)
- Analog Channel Mask (1 byte)
- Two bytes of digital data IF ANY DIGITAL CHANNELS ENABLED followed by...
- ...two bytes for EACH analog channel enabled...

• Q: How many bytes ATD02 ATD12 ATD23?

Readings and Assignments

- Readings
 - Building Wireless Sensor Networks, Chapter 4
- Assignments
 - Complete Doorbells
 - Romantic Lighting Sensor
 - Romantic Lighting Sensor with Feedback