Demo & Tutorial: images and audio transmission on wireless sensor networks

C. Pham
Winter School, RIIR, U. Oran
December, 10th, 2013
Oran, Algeria

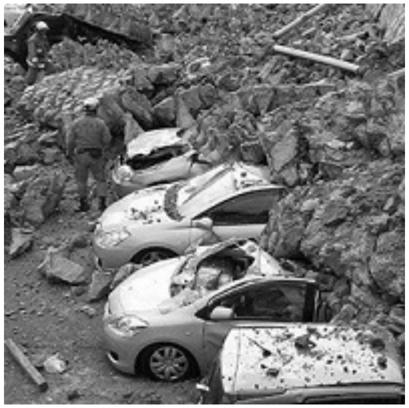
Prof. Congduc Pham
http://www.univ-pau.fr/~cpham
Université de Pau, France

Search & Rescue, Situation awareness

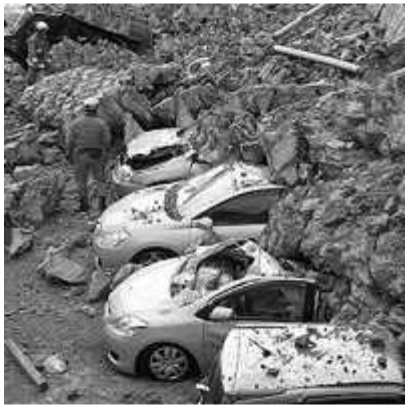
FDVision
Space Imaging 12/27/03

Dynamic Quality Factor 200x200

Original BMP 40000b

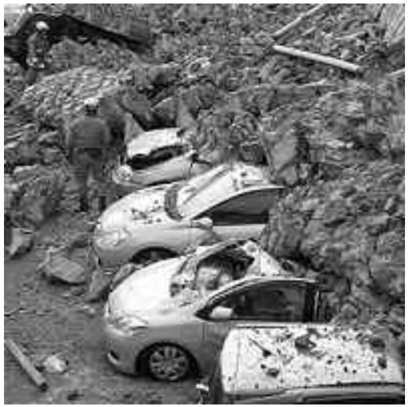


Q=50 S=11045b 142pkts



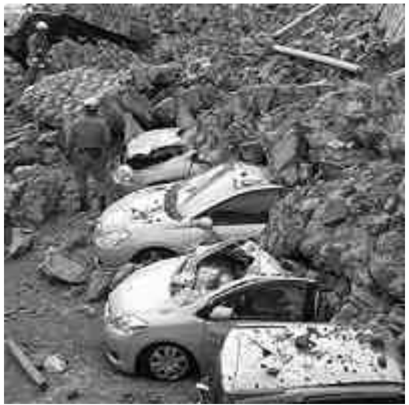
PSNR=25.1661

Q=40 S=9701b 123pkts



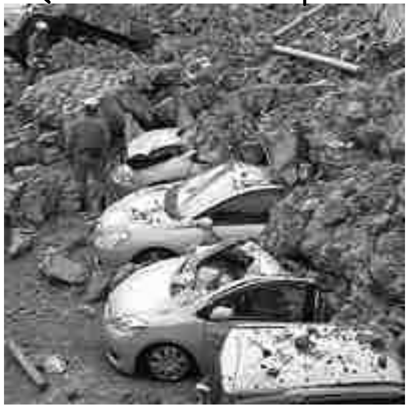
PSNR=24.2231

Q=30 S=8100b 101pkts



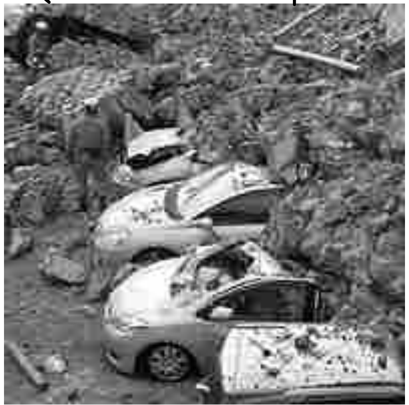
PSNR=23.2264

Q=20 S=6236b 76pkts



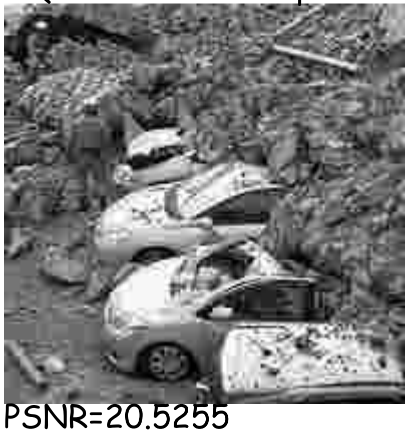
PSNR=22.1293

Q=15 S=5188b 63pkts



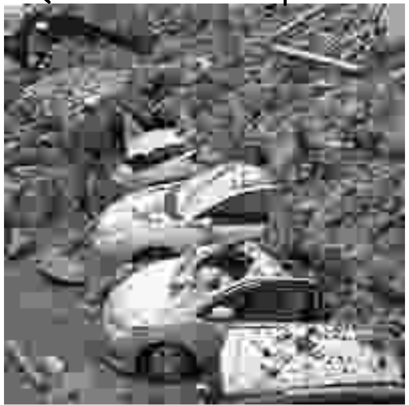
PSNR=21.4475

Q=10 S=3868b 47pkts



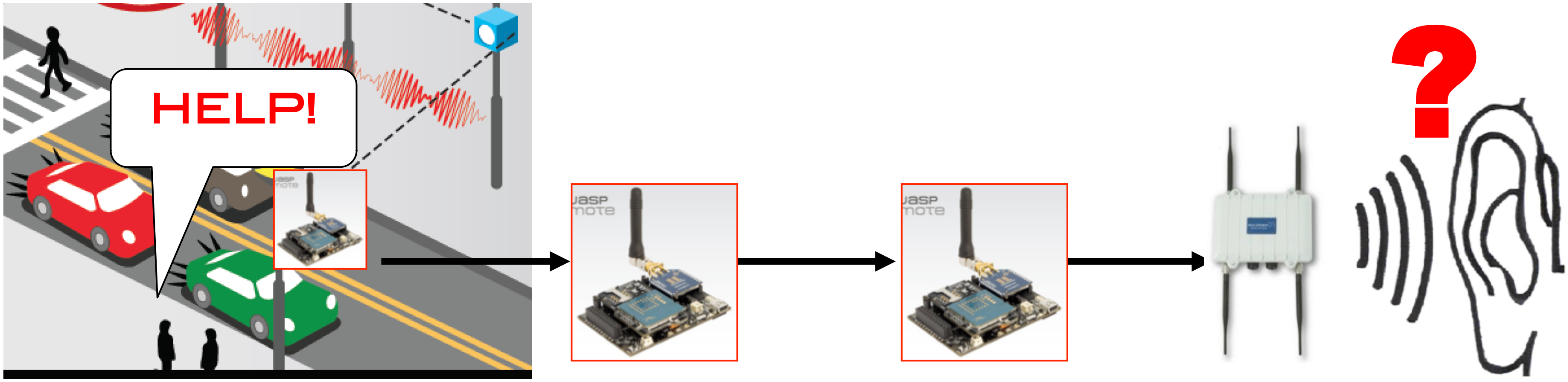
PSNR=20.5255

Q=5 S=2053b 24pkts



PSNR=18.937

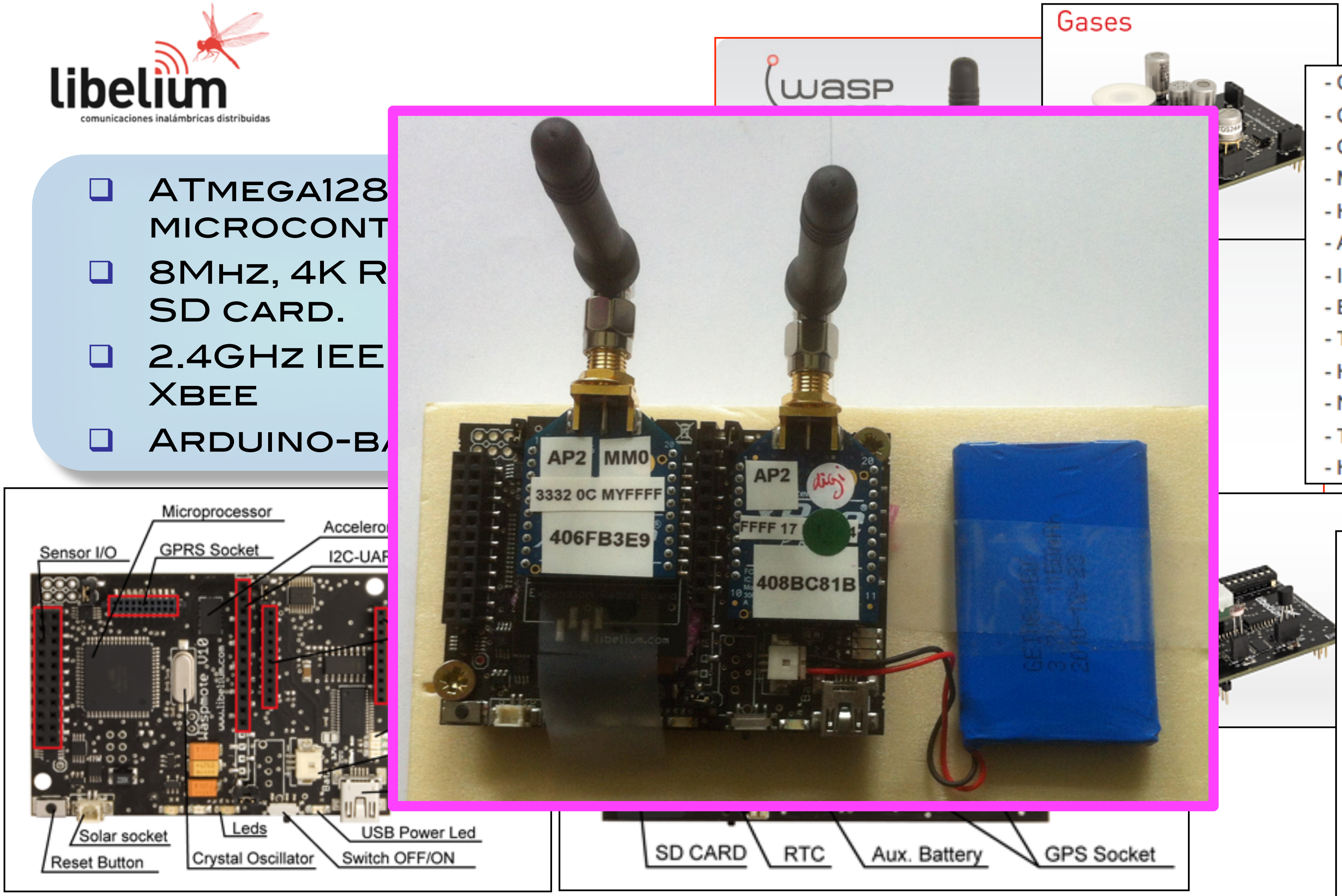
EAR-IT: audio surveillance in SmartCities and SmartBuildings



See http://www.ear-it.eu

SmartSantander IoT node

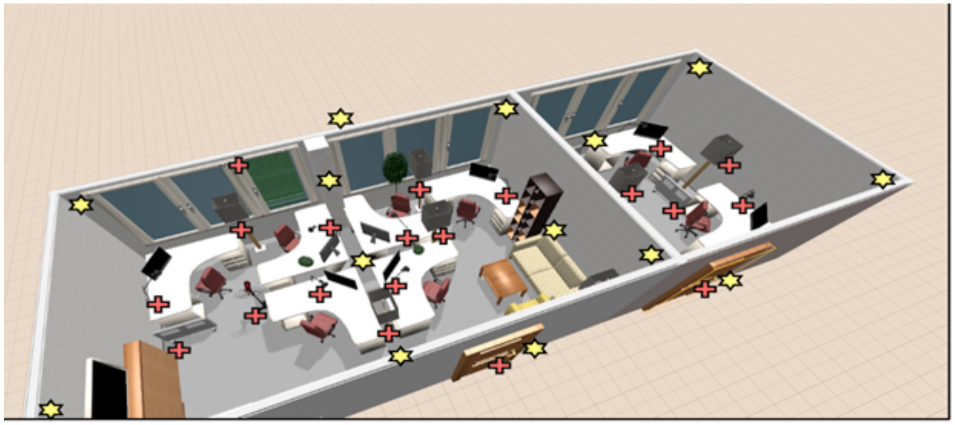
- ❑ ATmega128 microcont
- ❑ 8MHz, 4K R SD card.
- ❑ 2.4GHz IEE XBee
- ❑ Arduino-ba

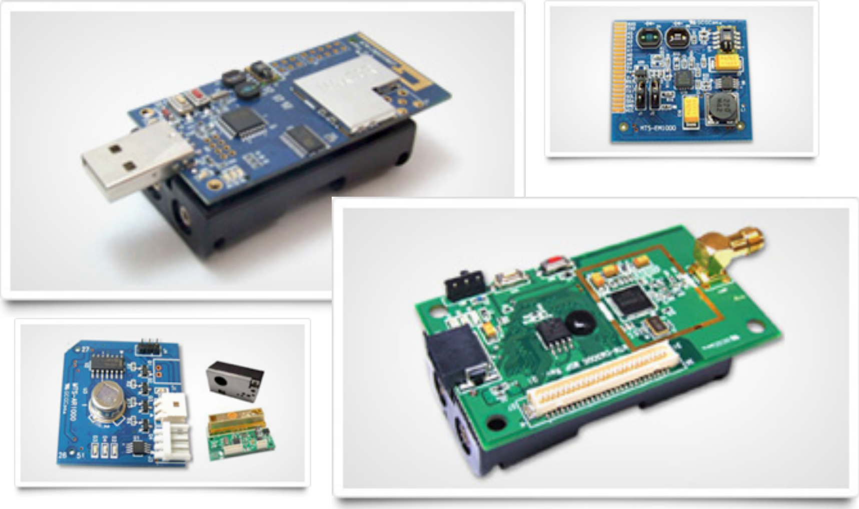


- Carbon Monoxide – CO
- Carbon Dioxide – CO2
- Oxygen – O2
- Methane – CH4
- Hydrogen – H2
- Ammonia – NH3
- Isobutane – C4H10
- Ethanol – CH3CH2OH
- Toluene – C6H5CH3
- Hydrogen Sulfide – H2S
- Nitrogen Dioxide – NO2
- Temperature
- Humidity

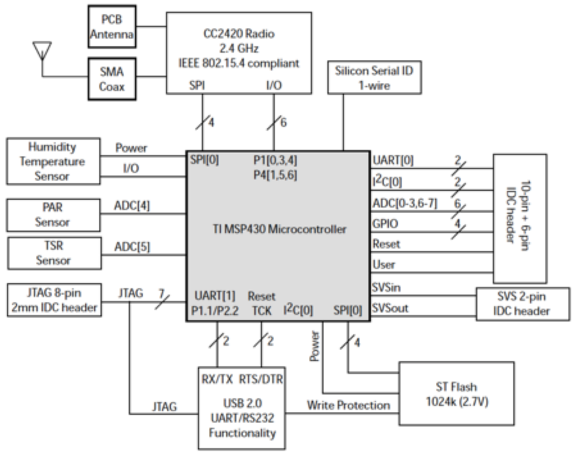
- Pressure/Weight
- Bend
- Vibration
- Impact
- Hall Effect
- Tilt
- Temperature (+/-)
- Liquid Presence
- Liquid Level
- Luminosity
- Presence (PIR)
- Stretch

HobNet test-bed at UNIGE

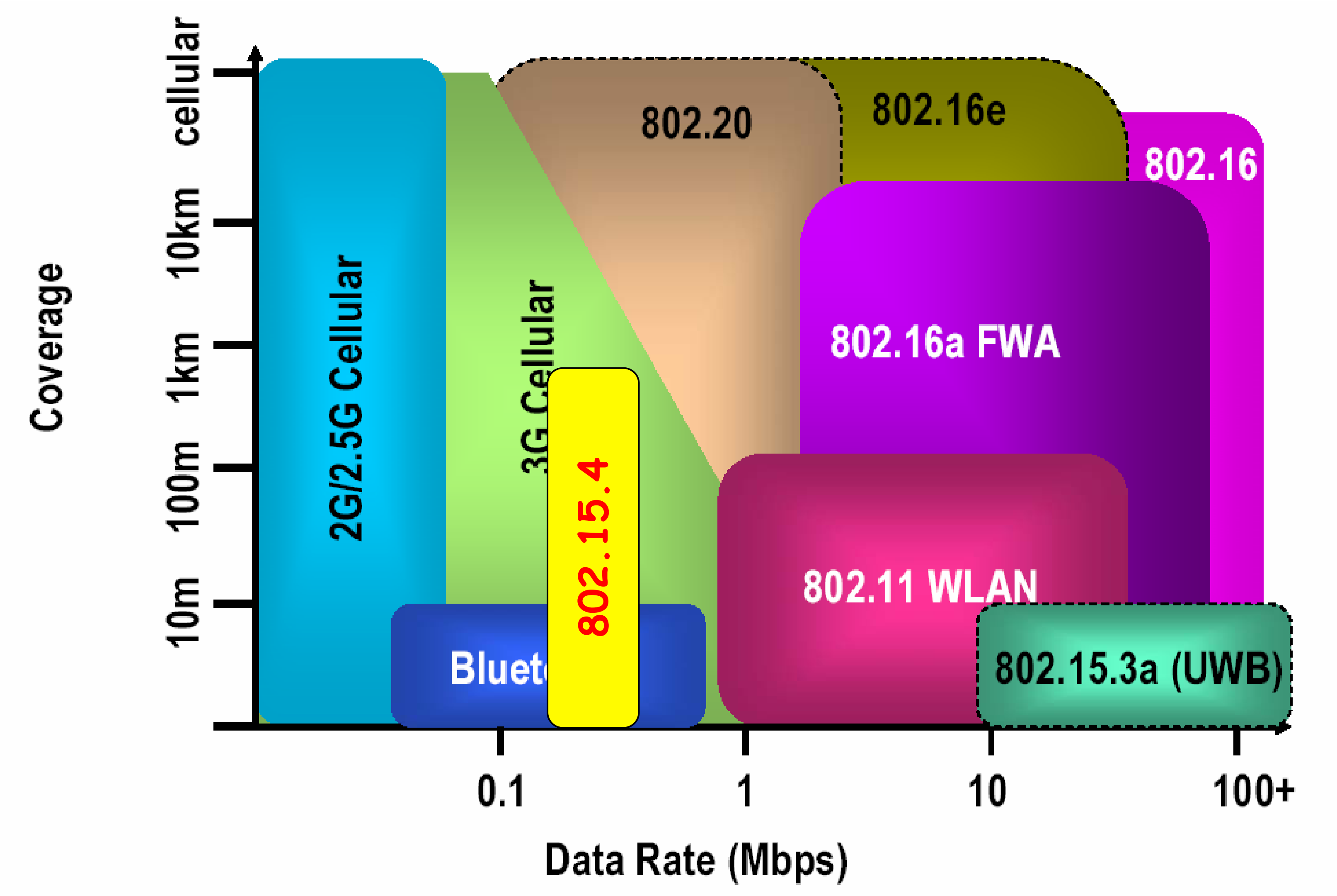




- MSP430F1611 microcontroller
- 8MHz, 48K flash, 10K RAM
- 2.4GHz IEEE 802.15.4 CC2420
- Programmed under TinyOS



Wireless technologies



IEEE 802.15.4

Caractéristiques Radio dans les réseaux de capteurs

- Norme ZigBee (IEEE 802.15.4 PHY)

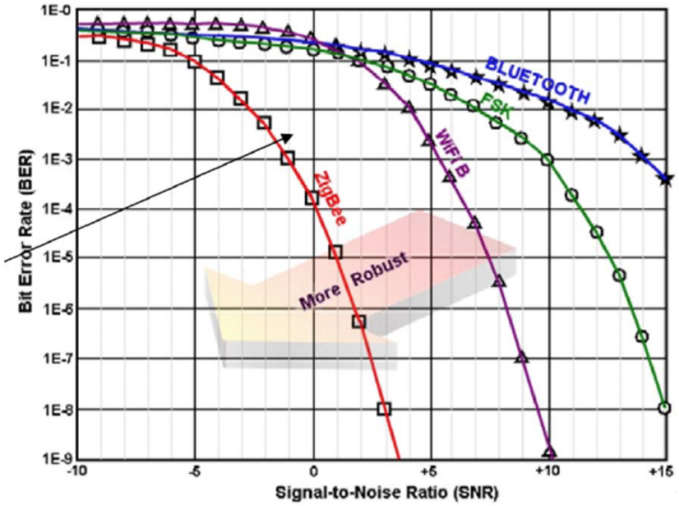
La norme IEEE802.15.4a, adaptées aux réseaux de capteurs, au contrôle industriel et aux dispositifs médicaux (CMI)

IEEE802.15.4 (couches 1 et 2):

□ Three bands, 27 channels specified

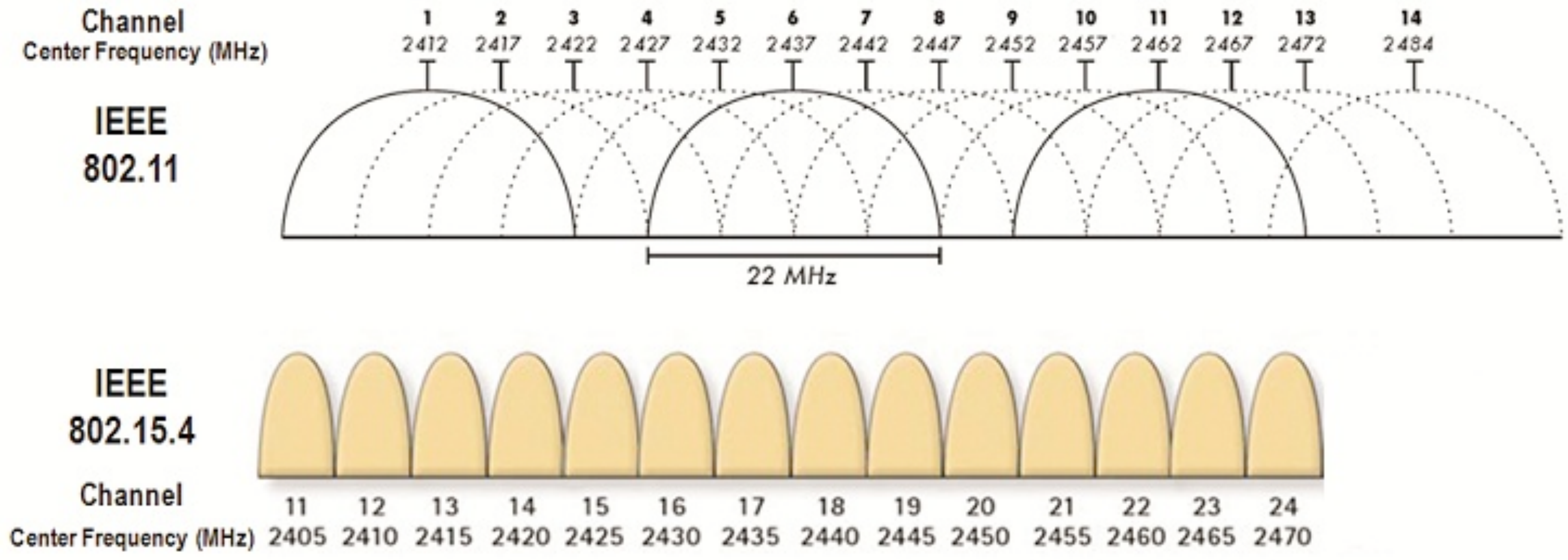
- 2.4 GHz: 16 channels, 250 kbps
- 868.3 MHz : 1 channel, 20 kbps
- 902-928 MHz: 10 channels, 40 kbps

Protocole	**Zigbee**	**Bluetooth**	**Wi-Fi**
IEEE	802.15.4	802.15.1	802.11a/b/g
Besoins mémoire	4-32 Kb	250 Kb +	1 Mb +
Autonomie avec pile	Années	Jours	Heures
Nombre de nœuds	65 000+	7	32
Vitesse de transfert	250 Kb/s	1 Mb/s	11-54 et + Mb/s
Portée	100 m	10-100 m	300 m



- Comparaison entre les normes ZigBee, Bluetooth et Wifi

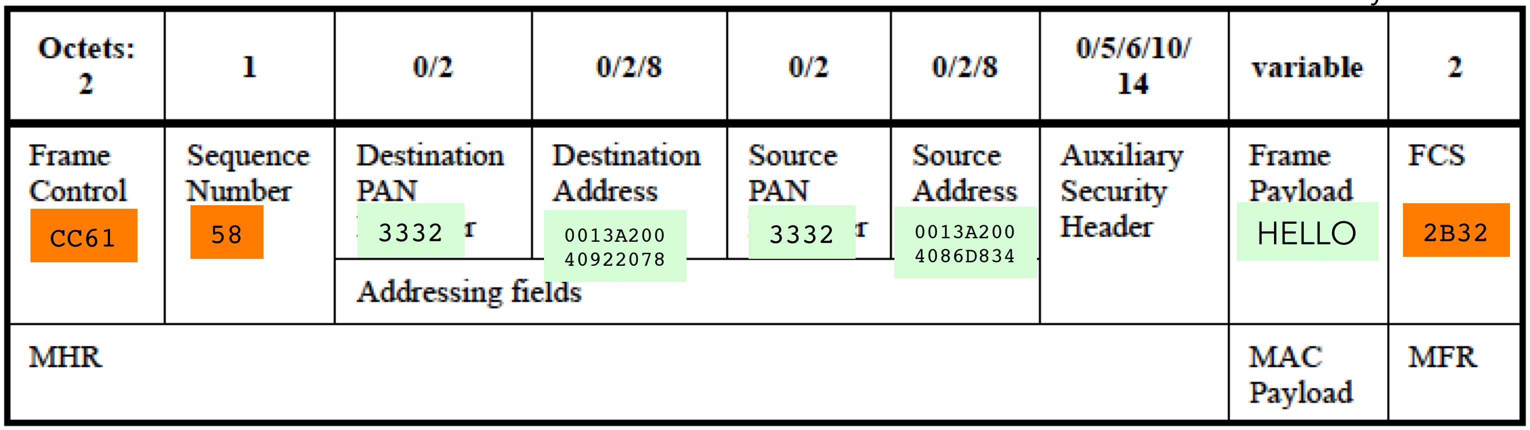
Spectrum band



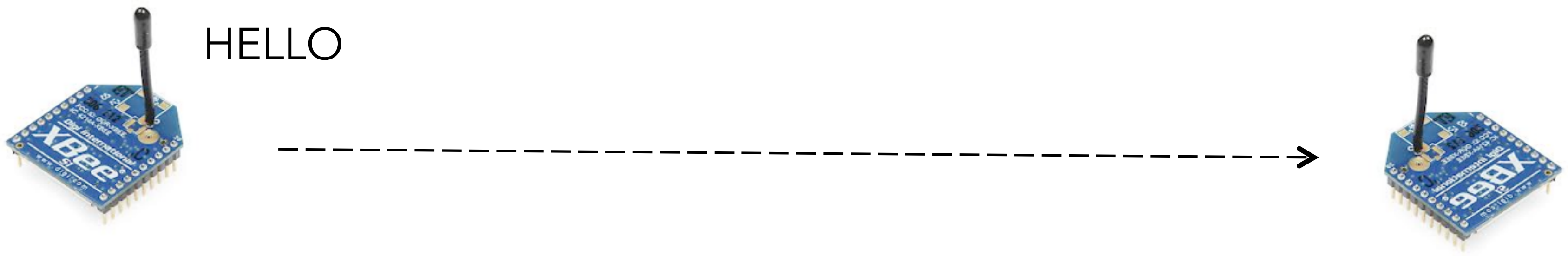
MAC FRAME FORMAT

Max size = 127 bytes

Max=102 bytes



Octets: 2	1	0/2	0/2/8	0/2	0/2/8	0/5/6/10/14	variable	2
Frame Control CC61	Sequence Number 58	Destination PAN 3332	Destination Address 0013A200 40922078	Source PAN 3332	Source Address 0013A200 4086D834	Auxiliary Security Header	Frame Payload HELLO	FCS 2B32
		Addressing fields						
MHR							MAC Payload	MFR



```
64-bit 0x0013A2004086D834
16-bit 0x0010
CHANNEL 0x0C
PANID 0x3332
```

```
64-bit 0x0013A20040922078
16-bit 0x0020
CHANNEL 0x0C
PANID 0x3332
```

802.15.4 GATEWAYS

Octets: 2	1	0/2	0/2/8	0/2	0/2/8	0/5/6/10/ 14	variable	2
Frame Control CC61	Sequence Number 58	Destination PAN 3332	Destination Address 0013A200 40922078	Source PAN 3332	Source Address 0013A200 4086D834	Auxiliary Security Header	Frame Payload HELLO	FCS 2B32
		Addressing fields						
MHR							MAC Payload	MFR

HELLO

```
64-bit 0x0013A2004086D834
16-bit 0x0010
PANID 0x3332
```

View as a serial port /dev/ttyUSB0

Transparent mode
Or Serial line
replacement mode

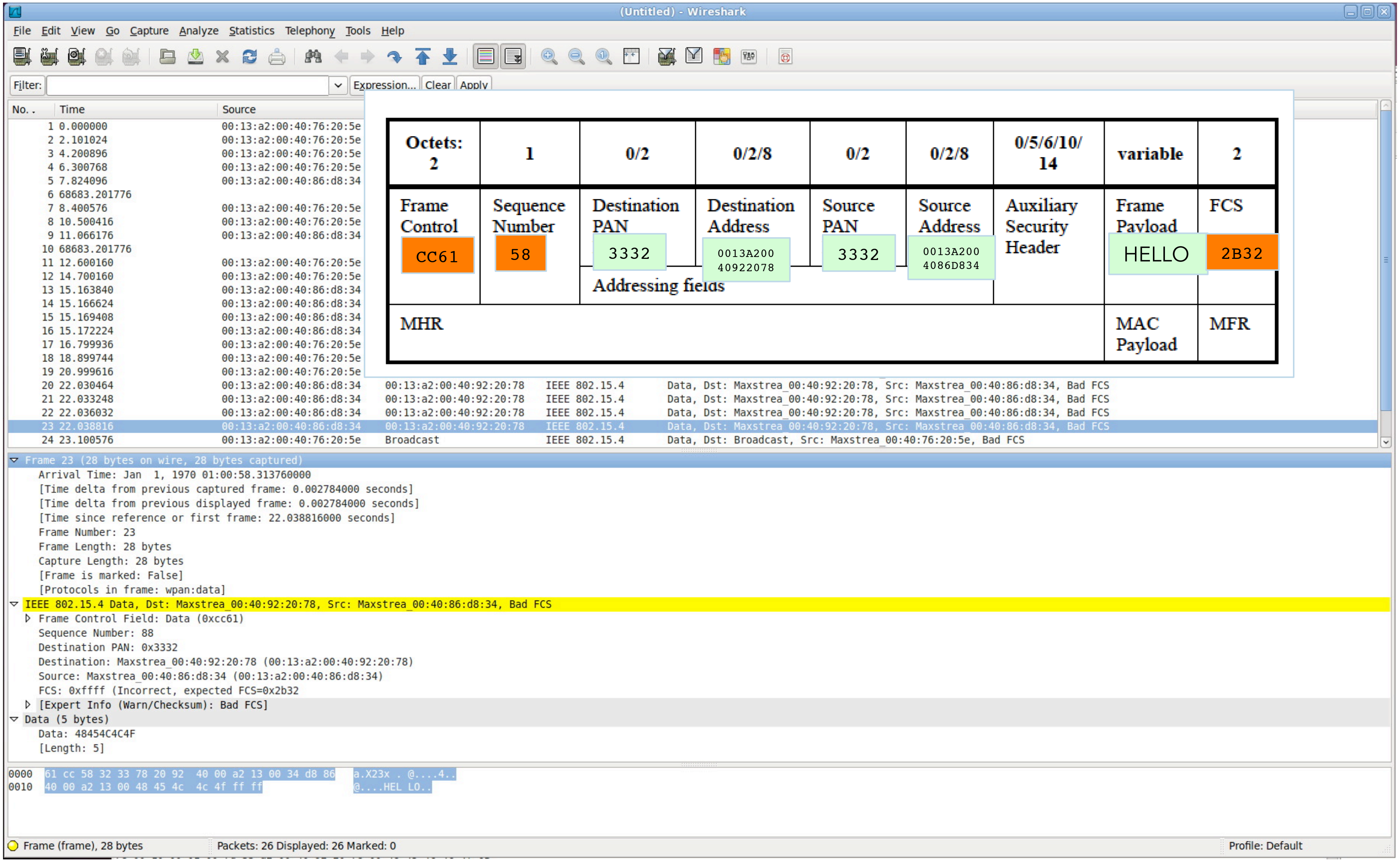
HELLO

Some hardware give access to
Link-layer information

CC61	58	3332	0013A200 40722098	3332	0013A200 4086D834	HELLO	2B32

HELLO

Radio sniffer



(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression... Clear Apply

Octets: 2	1	0/2	0/2/8	0/2	0/2/8	0/5/6/10/ 14	variable	2
Frame Control CC61	Sequence Number 58	Destination PAN 3332	Destination Address 0013A200 40922078	Source PAN 3332	Source Address 0013A200 4086D834	Auxiliary Security Header	Frame Payload HELLO	FCS 2B32
		Addressing fields						
MHR							MAC Payload	MFR

```
No. .  Time              Source
    1 0.000000           00:13:a2:00:40:76:20:5e
    2 2.101024           00:13:a2:00:40:76:20:5e
    3 4.200896           00:13:a2:00:40:76:20:5e
    4 6.300768           00:13:a2:00:40:76:20:5e
    5 7.824096           00:13:a2:00:40:86:d8:34
    6 68683.201776
    7 8.400576           00:13:a2:00:40:76:20:5e
    8 10.500416          00:13:a2:00:40:76:20:5e
    9 11.066176          00:13:a2:00:40:86:d8:34
   10 68683.201776
   11 12.600160          00:13:a2:00:40:76:20:5e
   12 14.700160          00:13:a2:00:40:76:20:5e
   13 15.163840          00:13:a2:00:40:86:d8:34
   14 15.166624          00:13:a2:00:40:86:d8:34
   15 15.169408          00:13:a2:00:40:86:d8:34
   16 15.172224          00:13:a2:00:40:86:d8:34
   17 16.799936          00:13:a2:00:40:76:20:5e
   18 18.899744          00:13:a2:00:40:76:20:5e
   19 20.999616          00:13:a2:00:40:76:20:5e
   20 22.030464          00:13:a2:00:40:86:d8:34  00:13:a2:00:40:92:20:78  IEEE 802.15.4   Data, Dst: Maxstrea_00:40:92:20:78, Src: Maxstrea_00:40:86:d8:34, Bad FCS
   21 22.033248          00:13:a2:00:40:86:d8:34  00:13:a2:00:40:92:20:78  IEEE 802.15.4   Data, Dst: Maxstrea_00:40:92:20:78, Src: Maxstrea_00:40:86:d8:34, Bad FCS
   22 22.036032          00:13:a2:00:40:86:d8:34  00:13:a2:00:40:92:20:78  IEEE 802.15.4   Data, Dst: Maxstrea_00:40:92:20:78, Src: Maxstrea_00:40:86:d8:34, Bad FCS
   23 22.038816          00:13:a2:00:40:86:d8:34  00:13:a2:00:40:92:20:78  IEEE 802.15.4   Data, Dst: Maxstrea_00:40:92:20:78, Src: Maxstrea_00:40:86:d8:34, Bad FCS
   24 23.100576          00:13:a2:00:40:76:20:5e  Broadcast                IEEE 802.15.4   Data, Dst: Broadcast, Src: Maxstrea_00:40:76:20:5e, Bad FCS
```

```
▽ Frame 23 (28 bytes on wire, 28 bytes captured)
    Arrival Time: Jan  1, 1970 01:00:58.313760000
    [Time delta from previous captured frame: 0.002784000 seconds]
    [Time delta from previous displayed frame: 0.002784000 seconds]
    [Time since reference or first frame: 22.038816000 seconds]
    Frame Number: 23
    Frame Length: 28 bytes
    Capture Length: 28 bytes
    [Frame is marked: False]
    [Protocols in frame: wpan:data]
▽ IEEE 802.15.4 Data, Dst: Maxstrea_00:40:92:20:78, Src: Maxstrea_00:40:86:d8:34, Bad FCS
  ▷ Frame Control Field: Data (0xcc61)
    Sequence Number: 88
    Destination PAN: 0x3332
    Destination: Maxstrea_00:40:92:20:78 (00:13:a2:00:40:92:20:78)
    Source: Maxstrea_00:40:86:d8:34 (00:13:a2:00:40:86:d8:34)
    FCS: 0xffff (Incorrect, expected FCS=0x2b32)
  ▷ [Expert Info (Warn/Checksum): Bad FCS]
▽ Data (5 bytes)
    Data: 48454C4C4F
    [Length: 5]
```

```
0000  61 cc 58 32 33 78 20 92  40 00 a2 13 00 34 d8 86    a.X23x . @....4..
0010  40 00 a2 13 00 48 45 4c  4c 4f ff ff                @....HEL LO..
```

Development environments

- Linux-based systems for higher flexibility and better interoperability
  - most of software tools are targeted for Unix
  - most of gateways devices are Linux-based (Meshlium, Beagle, Rasperry, ...)
- When possible, avoid Java development and priviledge C, or C++ and/or scripts (shell, python)

Standard software tools

- Libelium WaspMote
  - Libelium IDE (Arduino-based) & API development environment
- AdvanticSys TelosB
  - TinyOS 2.1.2 development environment
- Audio
  - Codec2 software (www.codec2.org): `c2enc`, `c2dec`
  - Speex software (www.speex.org): `speexenc`, `speexdec`
  - `sox` and `play` package (Linux)

Customized/dedicated tools

- Serial tools to read host computer serial port
  - `XBeeReceive` (C language)
  - `SerialToStdout` (python script)
- Communication tool to send command control packet
  - `XBeeSendCmd` (C language)
- To get a « pure » speex audio encoded file without any header
  - Modified version of `speexdec.c` (yes `speexdec.c` and not `speexenc.c`) compatible with Speex's `sampledec.c`
- Simple « pure » speex audio decoder without any header
  - Modified version of speex's `sampledec.c`: `speex_sampledec`

XBeeReceive

- XBeeReceive
  - Main target is XBee-based gateway
  - Translates XBee API frame
  - Read from the serial port
    - `/dev/ttyUSB0, /dev/ttyS0, …`
  - Display images in image mode
  - Reconstructs file in binary mode (handles packet losses)
    - Assumes each packet with 4 bytes header: 2 bytes for file size & 2 bytes for offset
  - Can write to Unix stdout & can act as a transparent serial replacement
  - Can act in a data stream fashion: no header for packets

XBeeReceive cmd line

```
USAGE:     ./XBeeReceive -baud b -p dev -onlydisplay img_file.dat -pktd -pktf -B/-I -ap0 -v val
                      —stdout -stream -Q 40 file_name
USAGE:     -baud, set baud rate, default is 38400
USAGE:     -p /dev/ttyUSB1
USAGE:     -onlydisplay img_file.dat, display the .dat file only
USAGE:     -pktd, display received XBee frames
USAGE:     -pktf, generate pkt list file
USAGE:     -B/-I, -B for binary mode, -I for image mode, default is image mode
USAGE:     -framing, expects 0xFF0x55 for binary mode, 0xFFx50 for image mode, default is no framing
USAGE:     -ap0, indicates an Xbee in AP mode 0 (transparent mode) so do not decode frame structure
USAGE:     -v 77, use 0x77 to fill in missing value in binary mode
USAGE:     -stdout, write to stdout for pipe mode, don't work with image mode
USAGE:     -stream, assumes no header & write to stdout for pipe mode in binary mode
USAGE:     -Q 40, use 40 as Quality Factor, default is 50
USAGE:     file_name, for images: give the original bmp file. for binary: give any file name
```

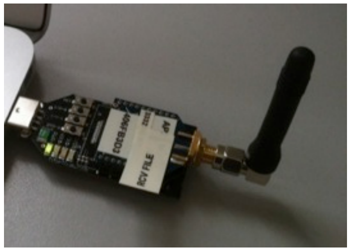
SerialToStdout.py

- Simple python script to read serial port when no translation is needed
- Change baud rate and port as needed

```
import serial
import sys

ser = serial.Serial('/dev/ttyUSB0', 38400, timeout=0)

# flush everything that may have been received on the port to make sure
# that we start with a clean serial input
ser.flushInput()

while True:
    out = ''
    sys.stdout.write(ser.read(1024))
    sys.stdout.flush()
```

- SerialToStdout.py can be use instead of XBeeReceive with an XBee in transparent mode

XBeeSendCmd



- XBeeSendCmd
  - Uses an XBee gateway to send ASCII string command, e.g. « /@D0030# »

```
USAGE:     ./XBeeSendCmd -p dev [-L][-DM][-at] -tinyos -tinyos_amid id_hex -mac|-net|-addr|-b message
USAGE:     -p /dev/ttyUSB1
USAGE:     -mac 0013a2004069165d HELLO
USAGE:     -net 5678 HELLO
USAGE:     -addr 64_or_16_bit_addr HELLO
USAGE:     -b HELLO
USAGE:     -at to send remote AT command: -at -mac 0013a2004069165d ATMM
USAGE:     -L insert Libelium API header
USAGE:     -DM to specify DigiMesh firmware
USAGE:     -tinyos to forge a TinyOS ActiveMessage compatible packet (0x3F0x05 are inserted)
USAGE:     -tinyos_amid 6F, to set the ActiveMessage identifier to 0x6F (0x05 is the default)
```

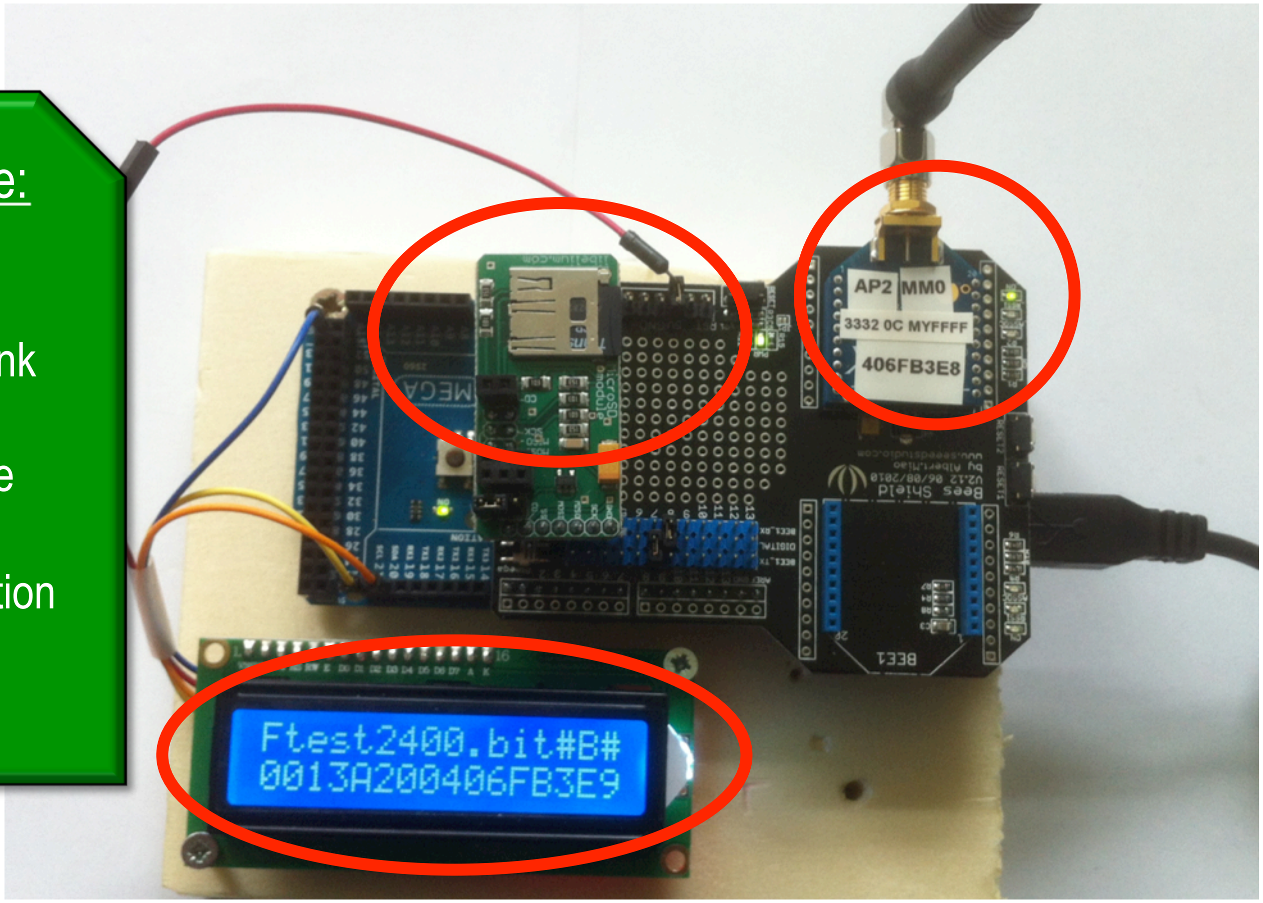
- Example:
  - `XBeeSendCmd –addr 0013A2004086D835 hello`
  - `XBeeSendCmd –addr 0013A2004086D835 /@Z50#`

Image demo

More generic solution: file sender node

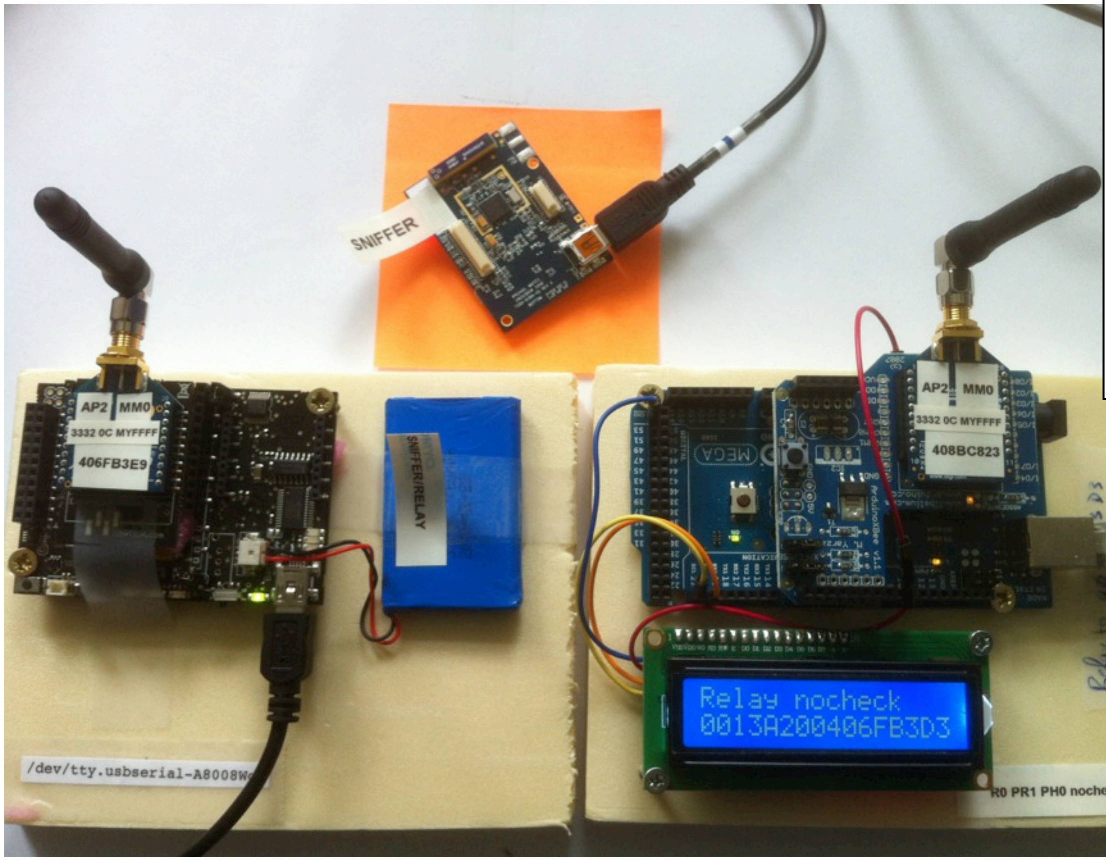
<u>Fully configurable:</u>

File to send
Size of packet chunk
Inter-packet delay
Image/Binary mode
Destination node
Clock synchronization



```
Ftest2400.bit#B#
0013A200406FB3E9
```

Relay nodes



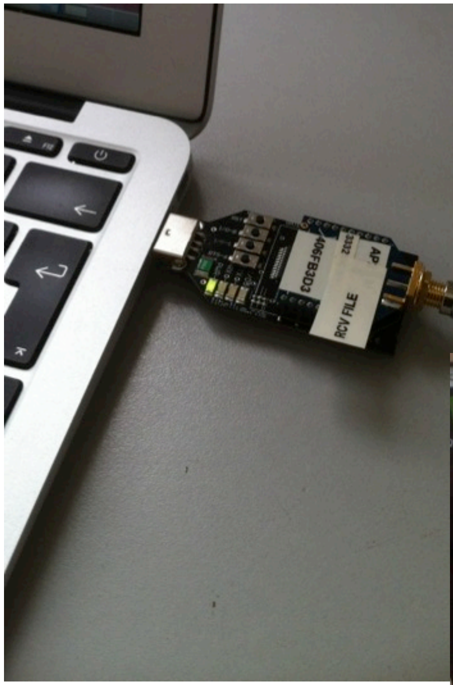
<u>Fully configurable:</u>

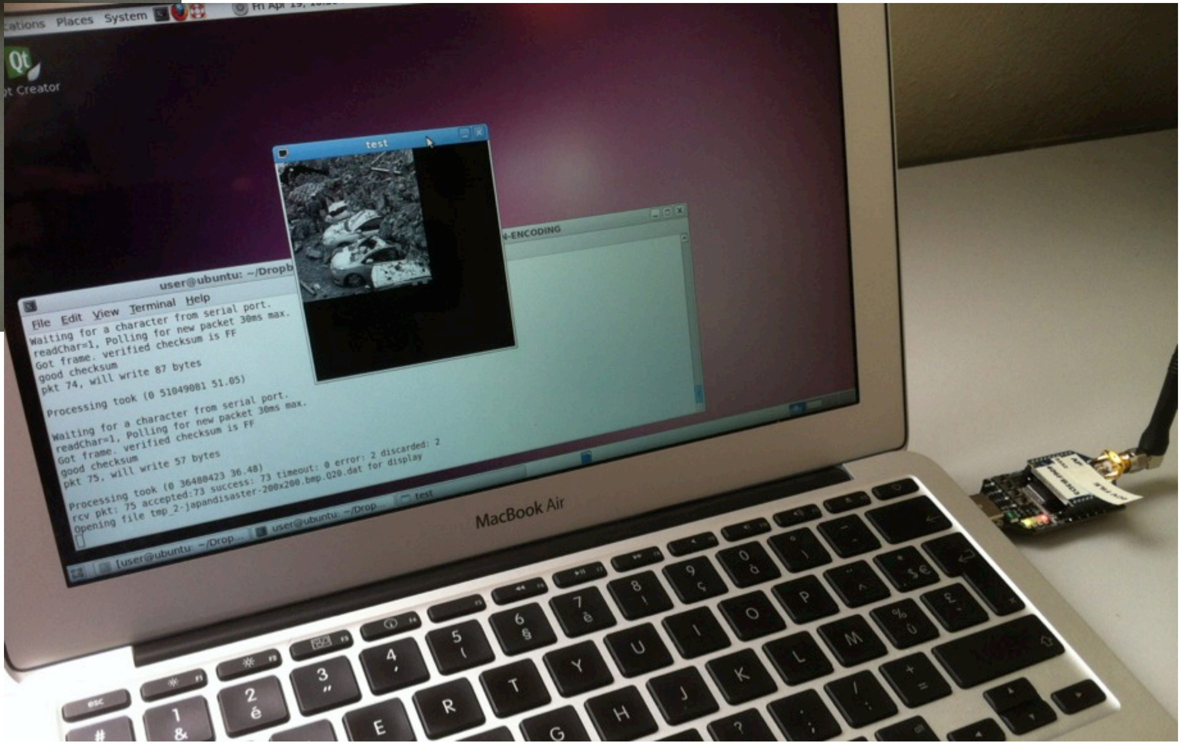
Destination node
Additional relay delay
Clock synchronization

Libelium WaspMote, Imote2, Arduino, TelosB, Micaz

Sink node

Linux PC/Laptop with USB/Serial gateway





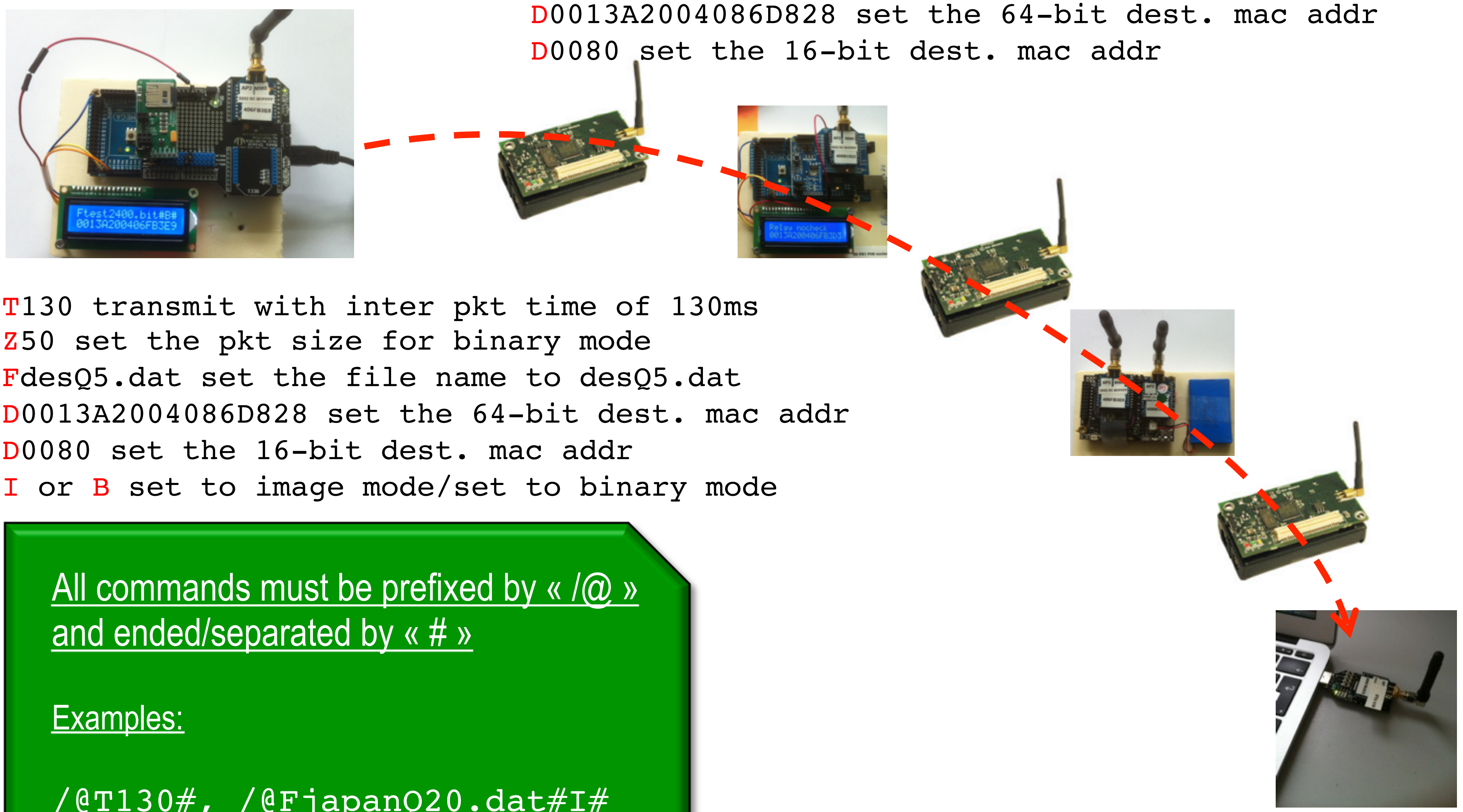
```
Waiting for a character from serial port.
readChar=1, Polling for new packet 30ms max.
Got frame. verified checksum is FF
good checksum
pkt 74, will write 87 bytes

Processing took (0 51049081 51.05)

Waiting for a character from serial port.
readChar=1, Polling for new packet 30ms max.
Got frame. verified checksum is FF
good checksum
pkt 75, will write 57 bytes

Processing took (0 36480423 36.48)
rcv pkt: 75 accepted:73 success: 73 timeout: 0 error: 2 discarded: 2
Opening file tmp_2-japandisaster-200x200.bmp.020.dat for display
```

Motivations

- ❑ Need a controlled environment to
  - ❑ Test multi-source scenario
  - ❑ Quantify impact of radio interference
  - ❑ Test multi-path routing and buffer management for congestion control
  - ❑ Know typical latencies
- ❑ Adopt a « fully controllable » approach
  - ❑ Each node can be dynamically configured…
  - ❑ … to « know » what is going on.

Mote nodes

R0/1 enable/disable relay mode
D0013A2004086D828 set the 64-bit dest. mac addr
D0080 set the 16-bit dest. mac addr



T130 transmit with inter pkt time of 130ms
Z50 set the pkt size for binary mode
FdesQ5.dat set the file name to desQ5.dat
D0013A2004086D828 set the 64-bit dest. mac addr
D0080 set the 16-bit dest. mac addr
I or B set to image mode/set to binary mode

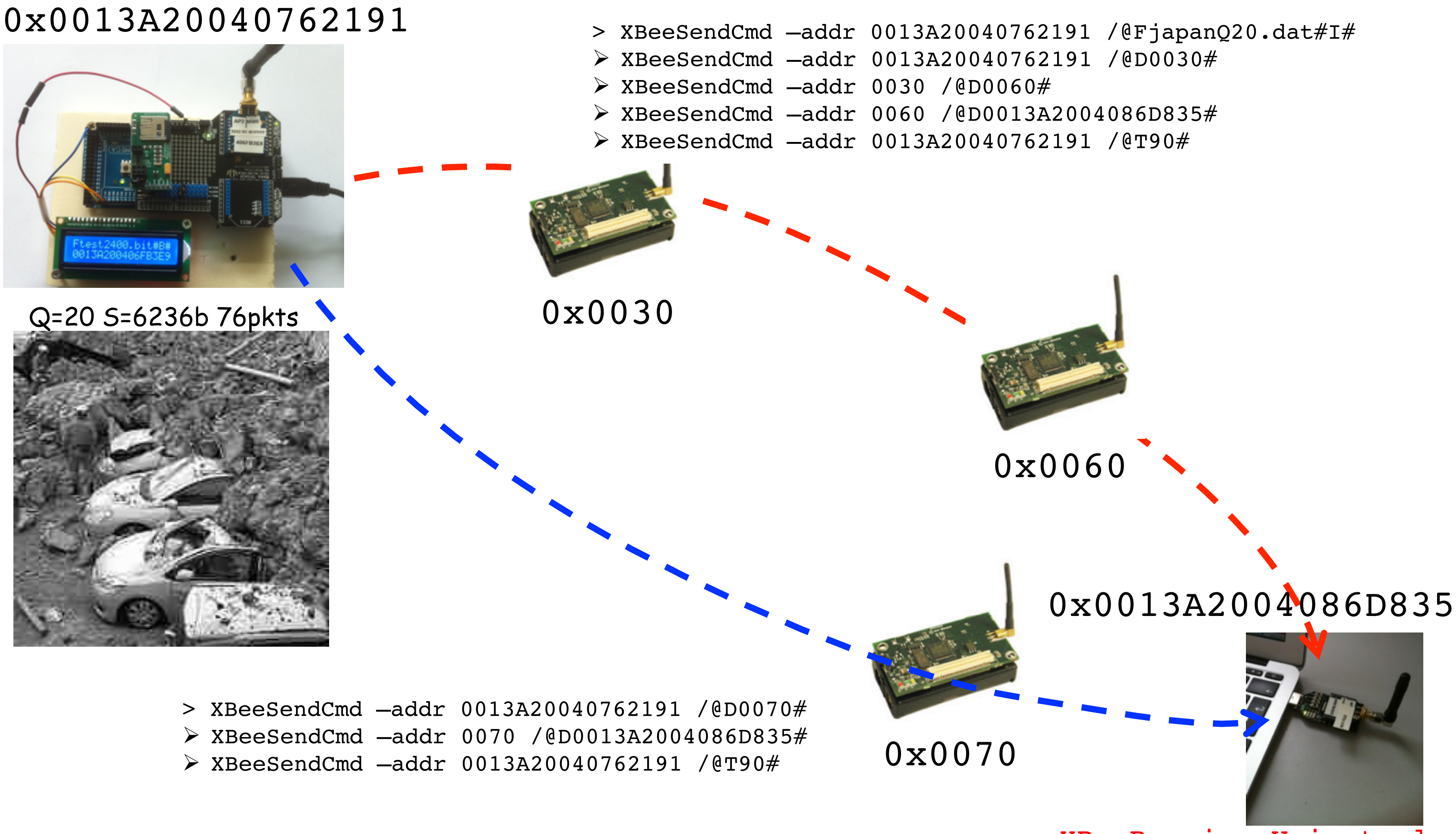
All commands must be prefixed by « /@ » and ended/separated by « # »

Examples:

/@T130#, /@FjapanQ20.dat#I#

XBeeReceive Unix tool

Image demo



0x0013A20040762191

Q=20 S=6236b 76pkts

```
> XBeeSendCmd —addr 0013A20040762191 /@FjapanQ20.dat#I#
➢ XBeeSendCmd —addr 0013A20040762191 /@D0030#
➢ XBeeSendCmd —addr 0030 /@D0060#
➢ XBeeSendCmd —addr 0060 /@D0013A2004086D835#
➢ XBeeSendCmd —addr 0013A20040762191 /@T90#
```

0x0030

0x0060

0x0013A2004086D835

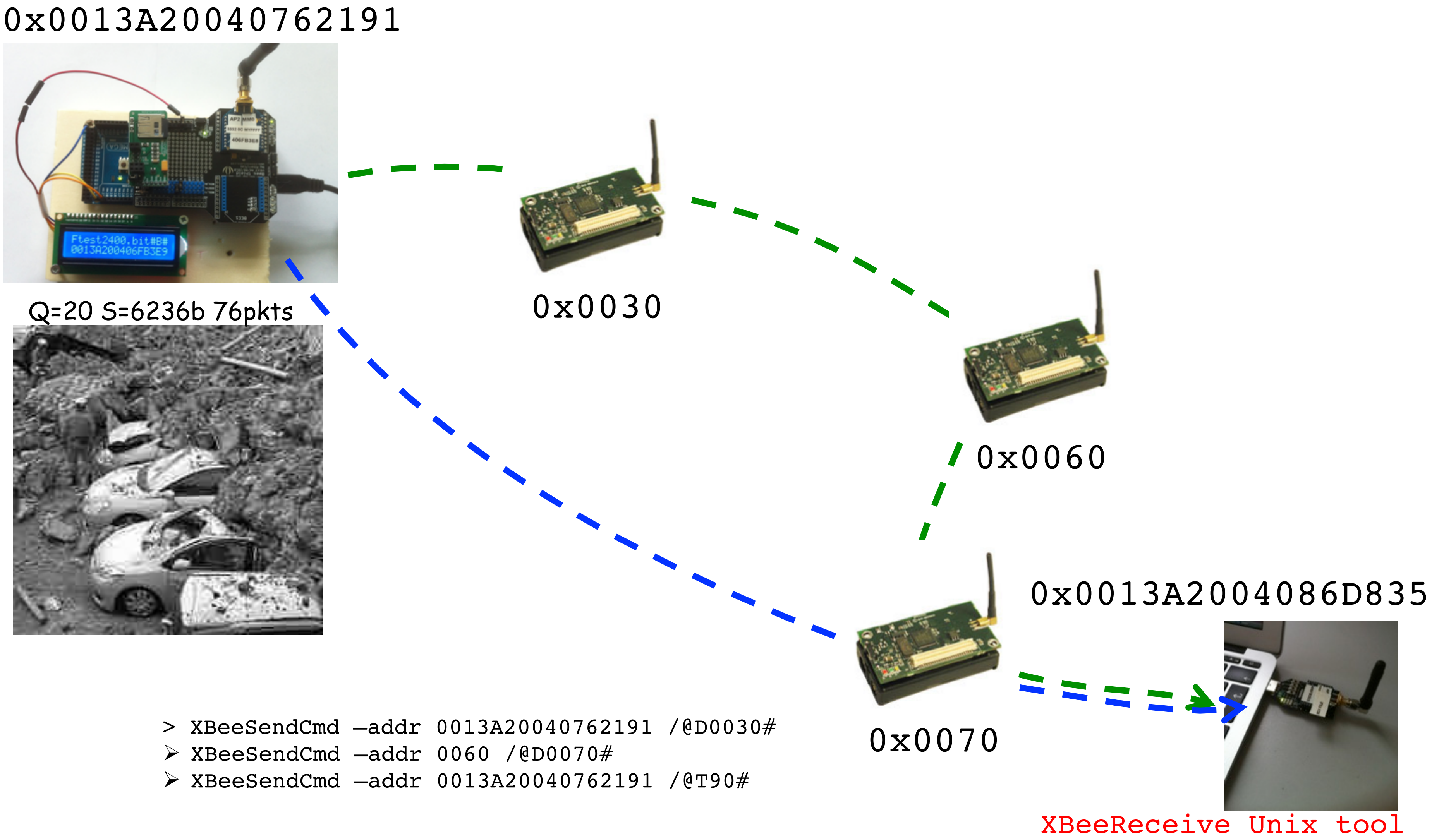
0x0070

```
> XBeeSendCmd —addr 0013A20040762191 /@D0070#
➢ XBeeSendCmd —addr 0070 /@D0013A2004086D835#
➢ XBeeSendCmd —addr 0013A20040762191 /@T90#
```

XBeeReceive Unix tool

```
➢ XBeeReceive —I —Q 20 japandisaster-200x200.bmp
```

Image demo



0x0013A20040762191

Q=20 S=6236b 76pkts

0x0030

0x0060

0x0070

0x0013A2004086D835

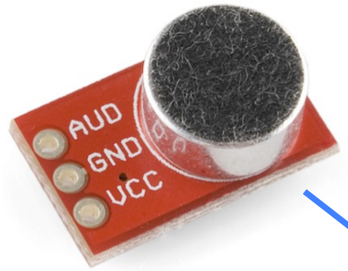
XBeeReceive Unix tool

```
> XBeeSendCmd –addr 0013A20040762191 /@D0030#
➢ XBeeSendCmd –addr 0060 /@D0070#
➢ XBeeSendCmd –addr 0013A20040762191 /@T90#
```

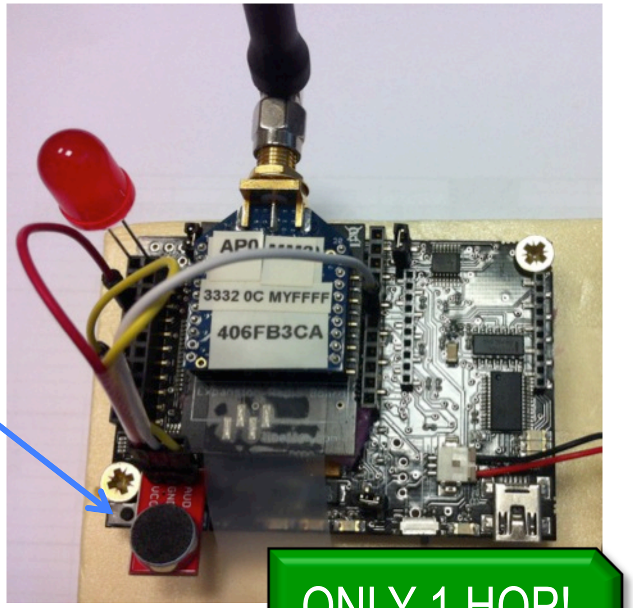
Audio demo

WaspMote+XBee in raw mode

- Electret mic with amplifier



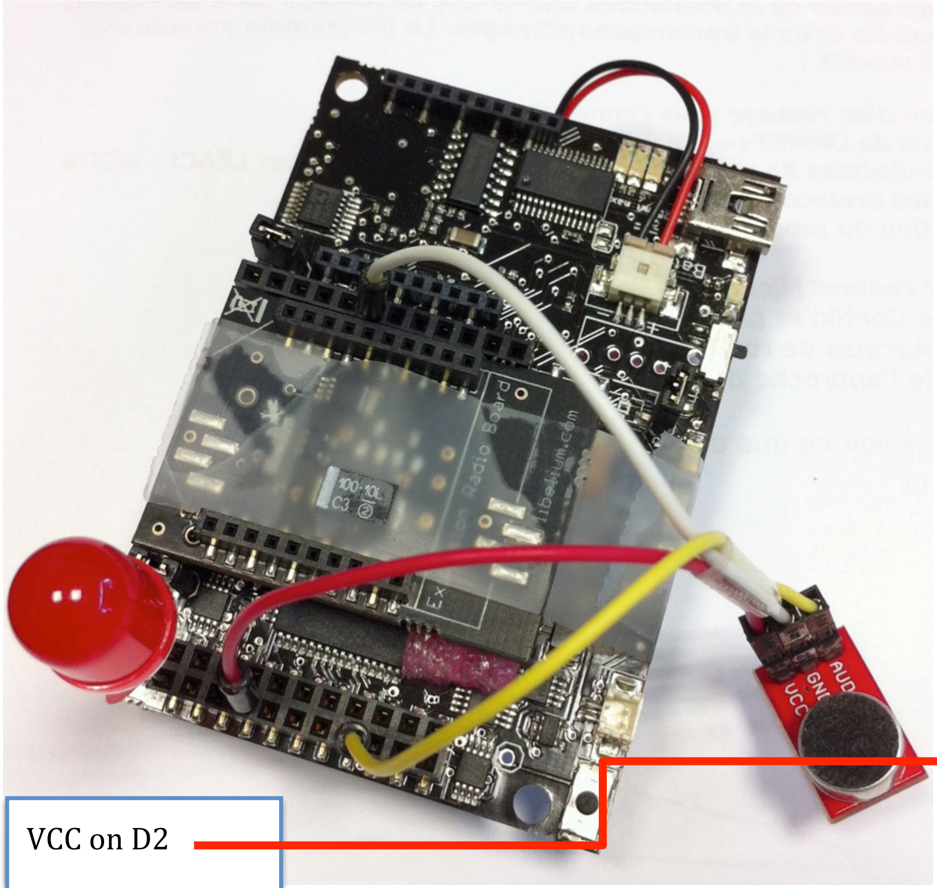
- XBee in AP0 mode (transparent mode)
- 8-bit 4Khz sampling gives 32000bps
- 8Khz sampling gives 64000bps, requires custom API

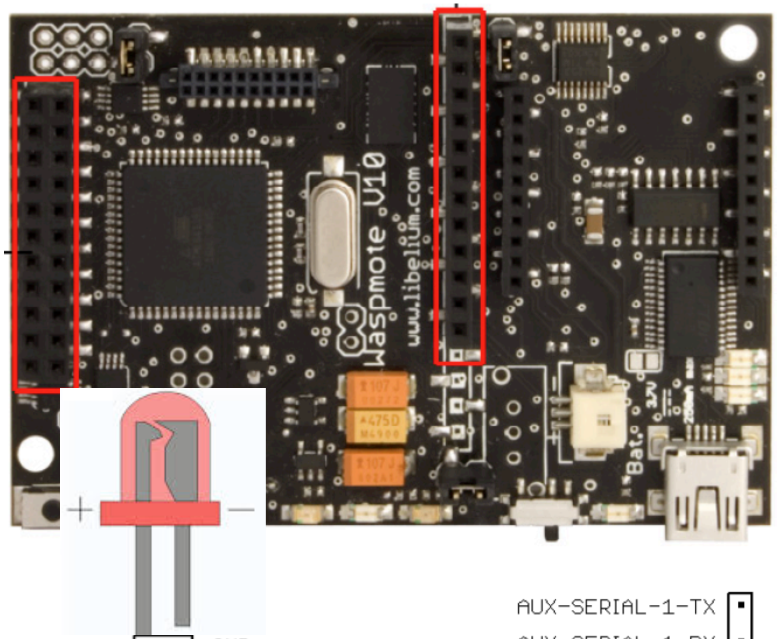


ONLY 1 HOP!

Xbee GW

Details of pin connection





DIGITAL8	GND
DIGITAL6	DIGITAL7
DIGITAL4	DIGITAL5
DIGITAL2	DIGITAL3
RESERVED	DIGITAL1
ANALOG6	ANALOG7
ANALOG4	ANALOG5
ANALOG2	ANALOG3
SENSOR POWER	ANALOG1
GPS POWER	5V SENSOR POWER
SDA	SCL

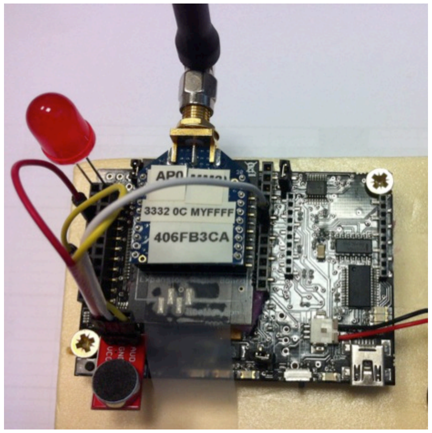
AUX-SERIAL-1-TX
AUX-SERIAL-1-RX
AUX-SERIAL-2-RX
AUX-SERIAL-2-TX
RESERVED
GND
GND
MUX_RX
MUX_TX
SENSOR POWER
SCL
SDA

VCC on D2

AUDIO on A2

GND on GND

WaspMote test-bed: XBee gw AP0



```
void loop() {
        val = analogRead(ANALOG2); // read analog value
        val8bit = ((val >> 2) ); // convert into 8 bit
        // write on UART1, need an XBee module
        // with AP mode 0

        serialWrite(val8bit,1);
}
```

Xbee GW

With XBee GW also in AP0 mode

```
4KHz sampling
> XBeeReceive -baud 38400 -ap0 -stdout dumb.dat | play --buffer 50 -t raw –r 4000 -u -1 –

8KHz sampling
> XBeeReceive -baud 125000 -ap0 -stdout dumb.dat | play --buffer 50 -t raw -r 8000 -u -1 -

Save raw data for off-line playing
> XBeeReceive -baud 38400 -ap0 -stdout dumb.dat > test.raw
> play -t raw –r 4000 -u -1 test.raw
```

Alternatively using SerialToStdout python script, at 38400 baud only

```
> python SerialToStdout | play --buffer 50 -t raw –r 4000 -u -1 –
```

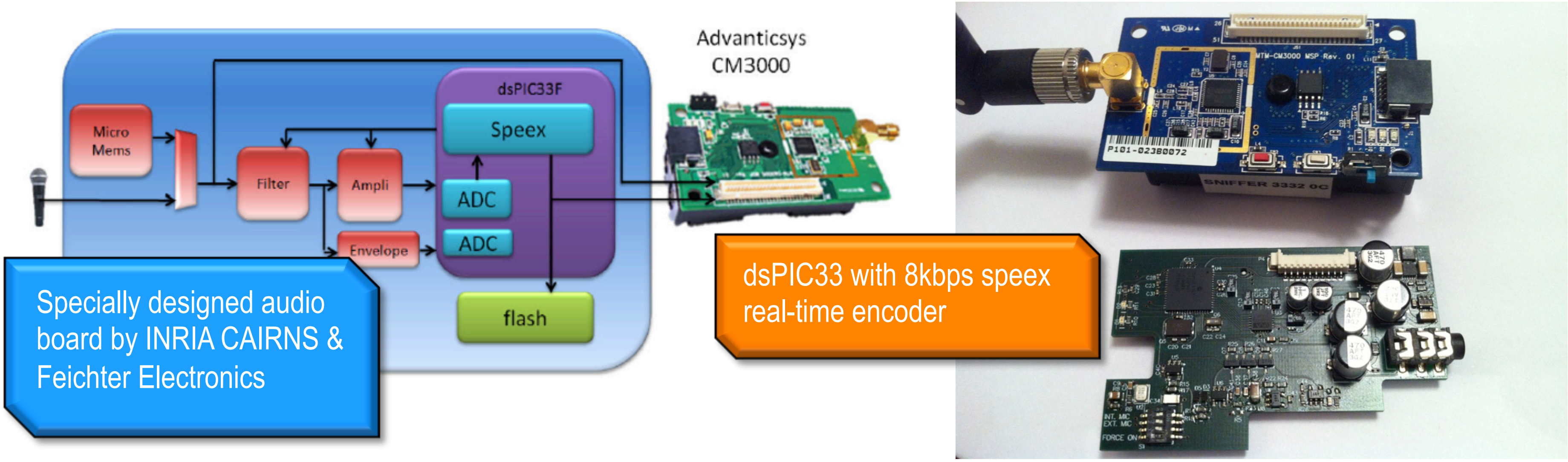
XBee gateway in pkt mode (AP2)

- The receiving XBee module may need to be in packet mode (AP2) due to deployment constraints
- Adds overhead of XBee API frame decoding: 8KHz sampling may be not supported

```
4KHz sampling
> XBeeReceive -baud 38400 —stream dumb.dat | play --buffer 50 -t raw —r 4000 -u -1 —


Save raw data for off-line playing
> XBeeReceive -baud 38400 —stream dumb.dat > test.raw
> play -t raw —r 4000 -u -1 test.raw
```

Multi-hop audio solution

- Use dedicated audio board for sampling/storing/encoding at 8kbps

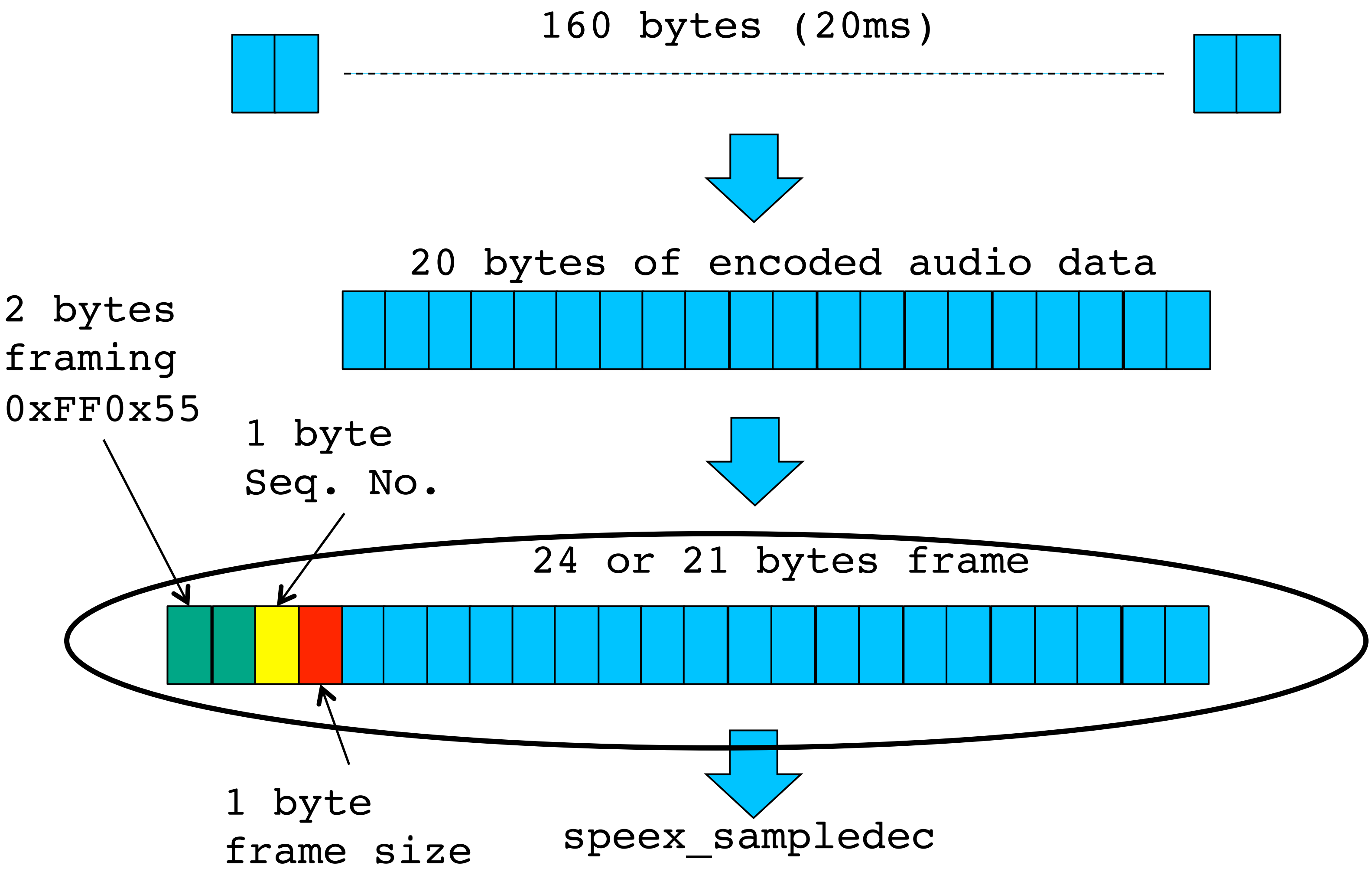


Specially designed audio board by INRIA CAIRNS & Feichter Electronics

dsPIC33 with 8kbps speex real-time encoder

- Allows for multi-hop, encoded audio streaming scenarios

speex at 8kbps



160 bytes (20ms)

20 bytes of encoded audio data

2 bytes framing 0xFF0x55

1 byte Seq. No.

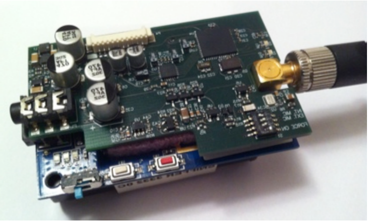
24 or 21 bytes frame

1 byte frame size

speex_sampledec

1-hop test-bed w/audio board

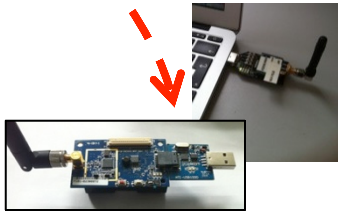
0x0090



SPEEX AUDIO ENCODING
8KBPS

```
A1/2/3/4 aggregate audio frames
D0100 set the 16-bit dest. mac addr
C0/1 power off/on the audio board
```

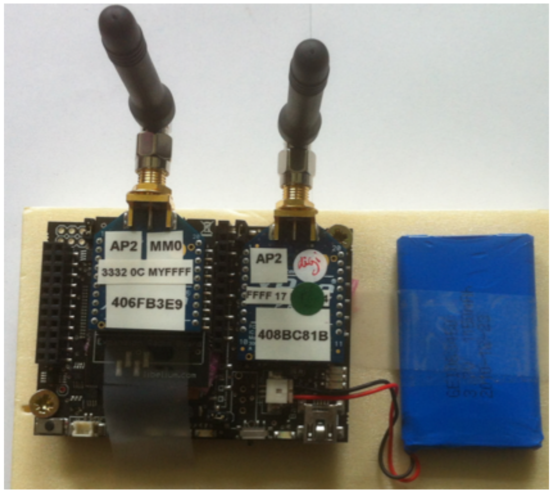
0x0100



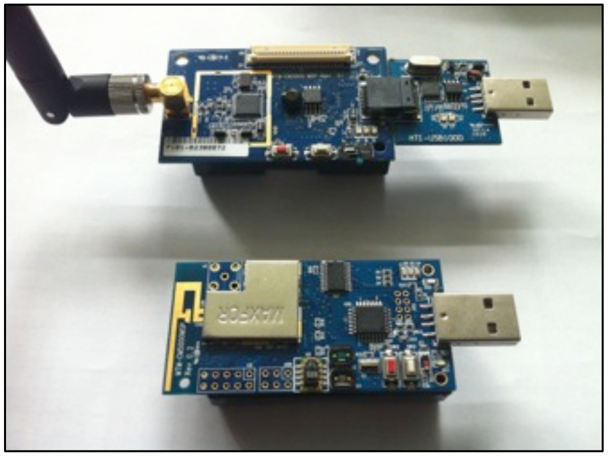
DECODE & PLAY
RECEIVED AUDIO

```
python 115200SerialToStdout.py | ./speex_sampledec_wframing essai.raw |
play --buffer 100 -t raw -r 8000 -s -2 -
```

Relay nodes



Libelium
WaspMote



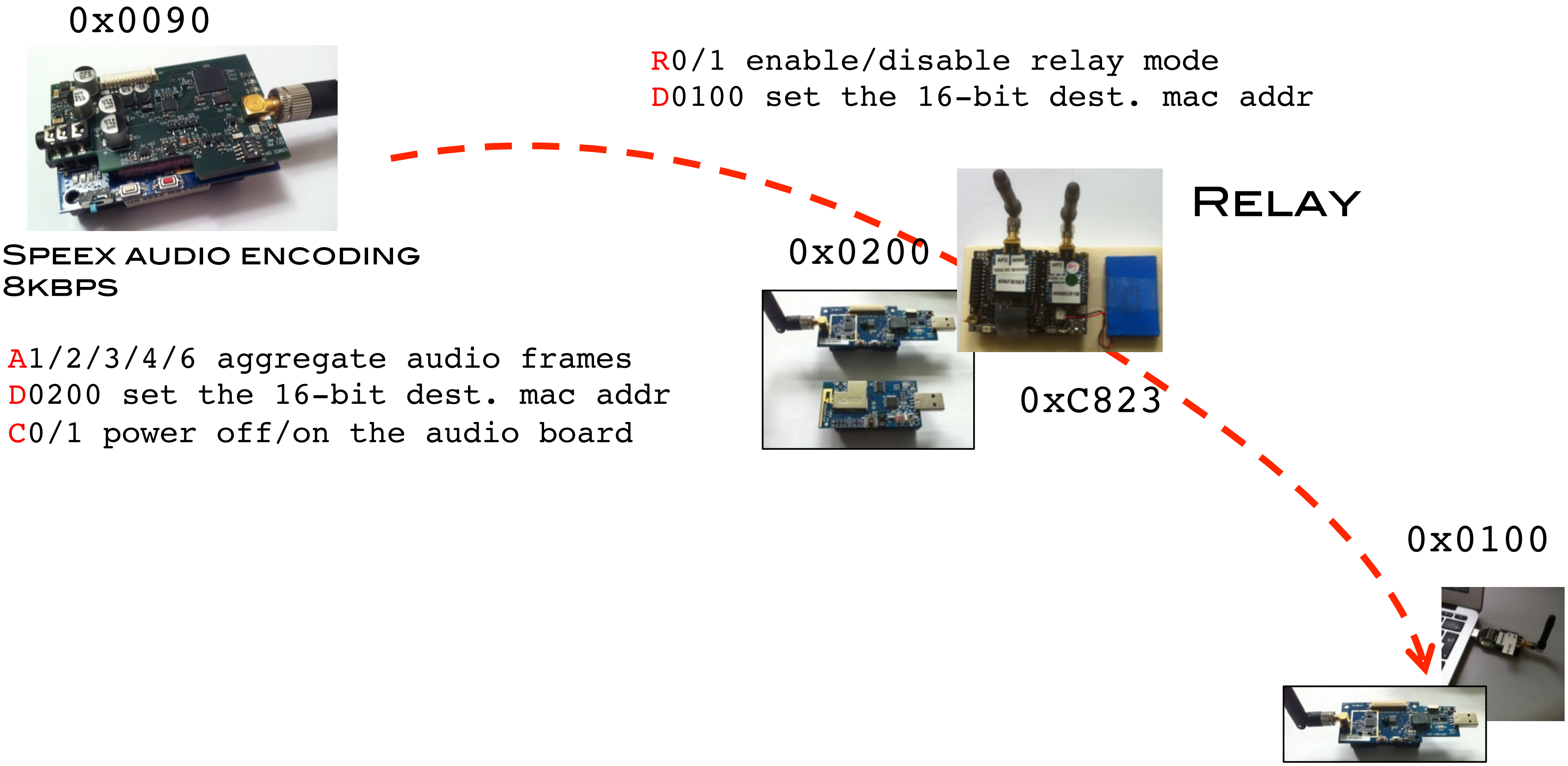
AdvanticSys
CM5000, CM3000

Fully configurable:

Destination node
Additional relay delay
Clock synchronization

```
R0/1 enable/disable relay mode
D0013A2004086D828 set the 64-bit dest. mac addr
D0080 set the 16-bit dest. mac addr
```

2-hop test-bed w/audio board



0x0090

SPEEX AUDIO ENCODING
8KBPS

A1/2/3/4/6 aggregate audio frames
D0200 set the 16-bit dest. mac addr
C0/1 power off/on the audio board

R0/1 enable/disable relay mode
D0100 set the 16-bit dest. mac addr

0x0200

RELAY

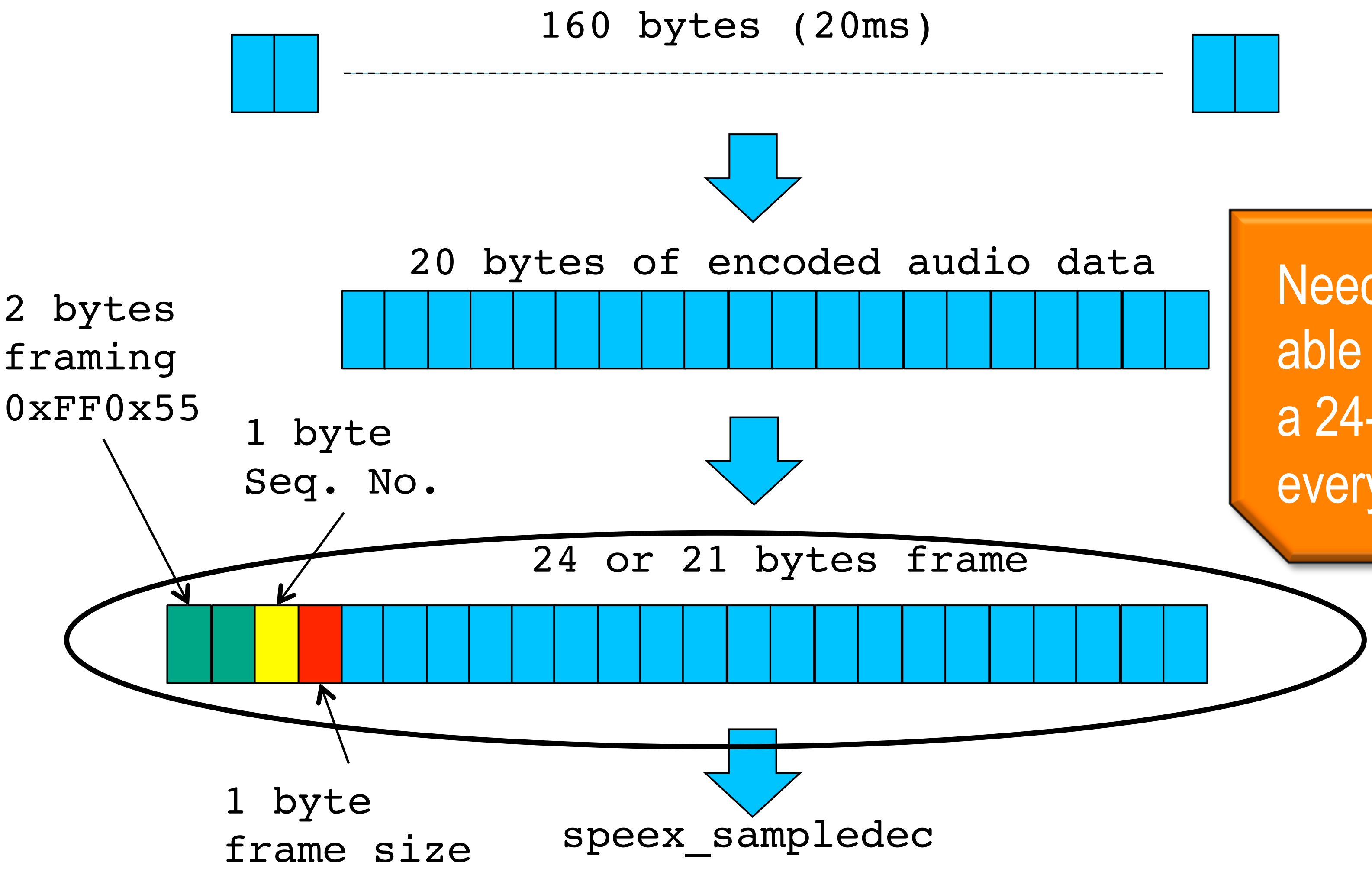
0xC823

0x0100

```
python 115200SerialToStdout.py | ./speex_sampledec_wframing essai.raw |
play --buffer 100 -t raw -r 8000 -s -2 -
```

DECODE & PLAY
RECEIVED AUDIO

speex at 8kbps requirements



160 bytes (20ms)

20 bytes of encoded audio data

2 bytes framing 0xFF0x55

1 byte Seq. No.

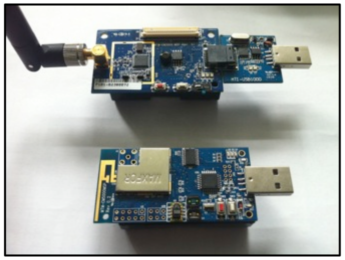
24 or 21 bytes frame

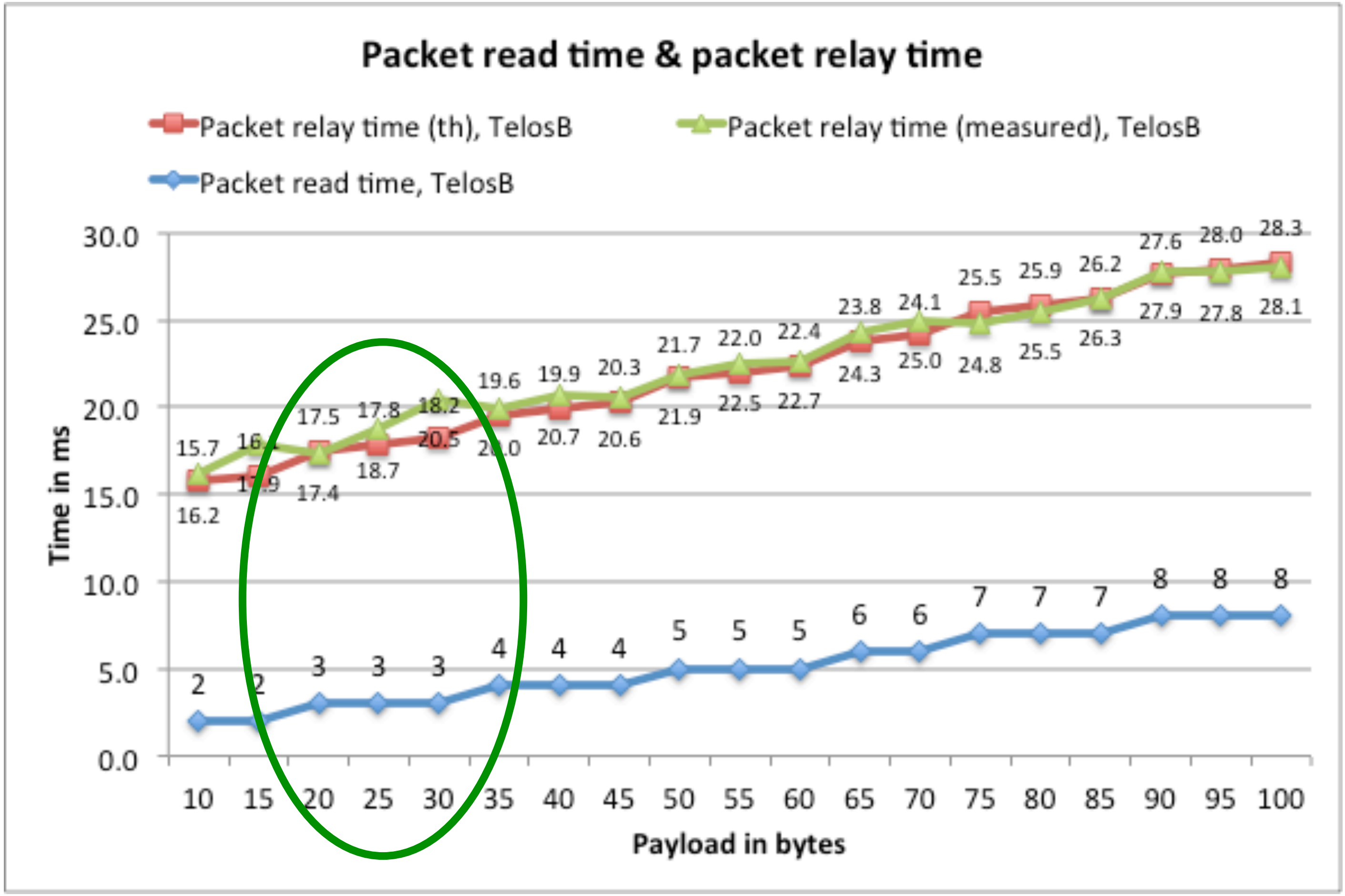
1 byte frame size

speex_sampledec

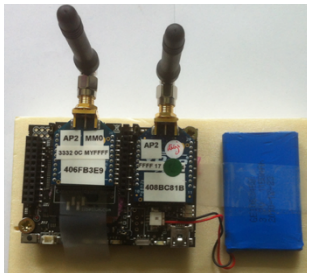
Need to be able to relay a 24-byte pkt every 20ms

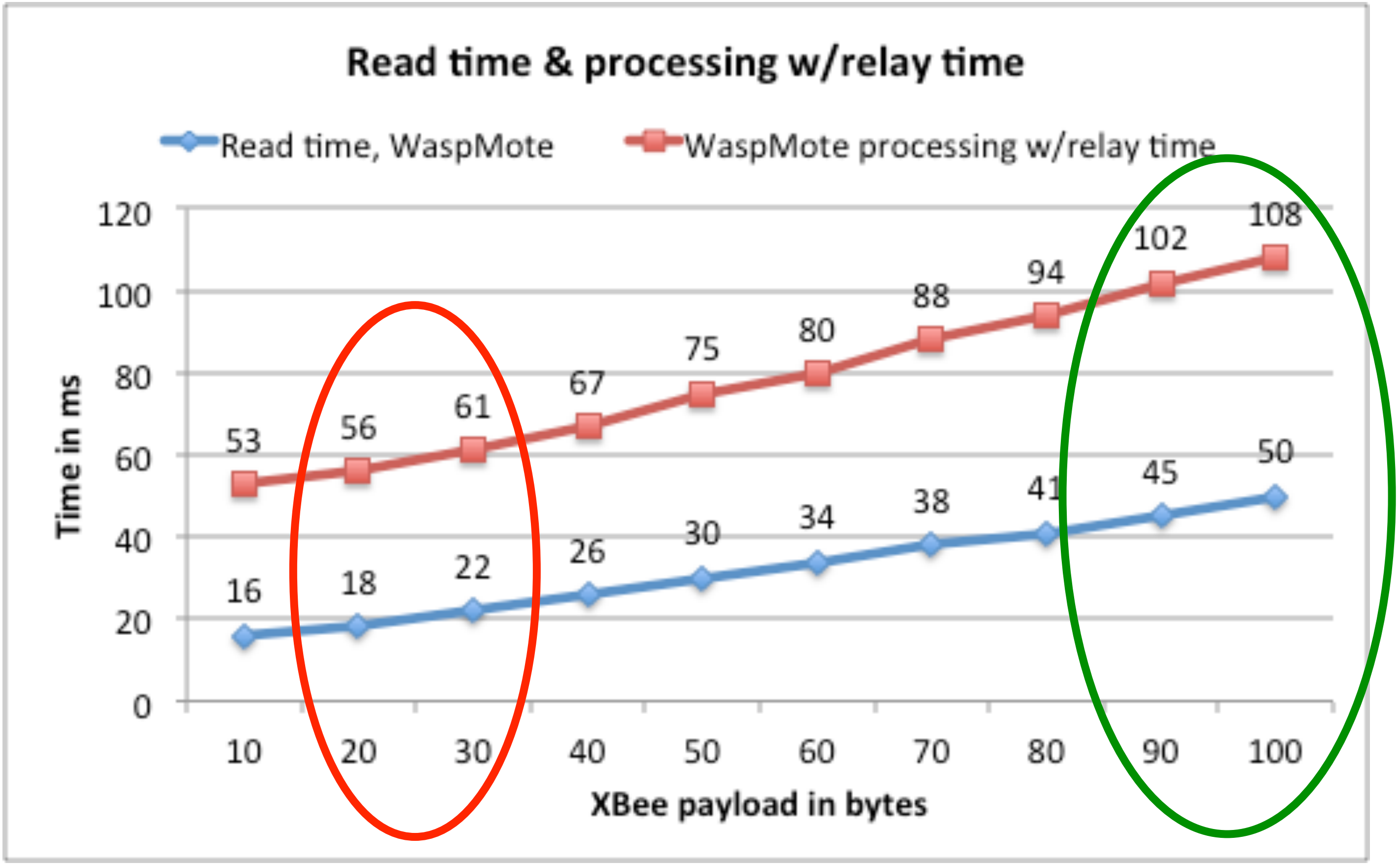
Relay node performances



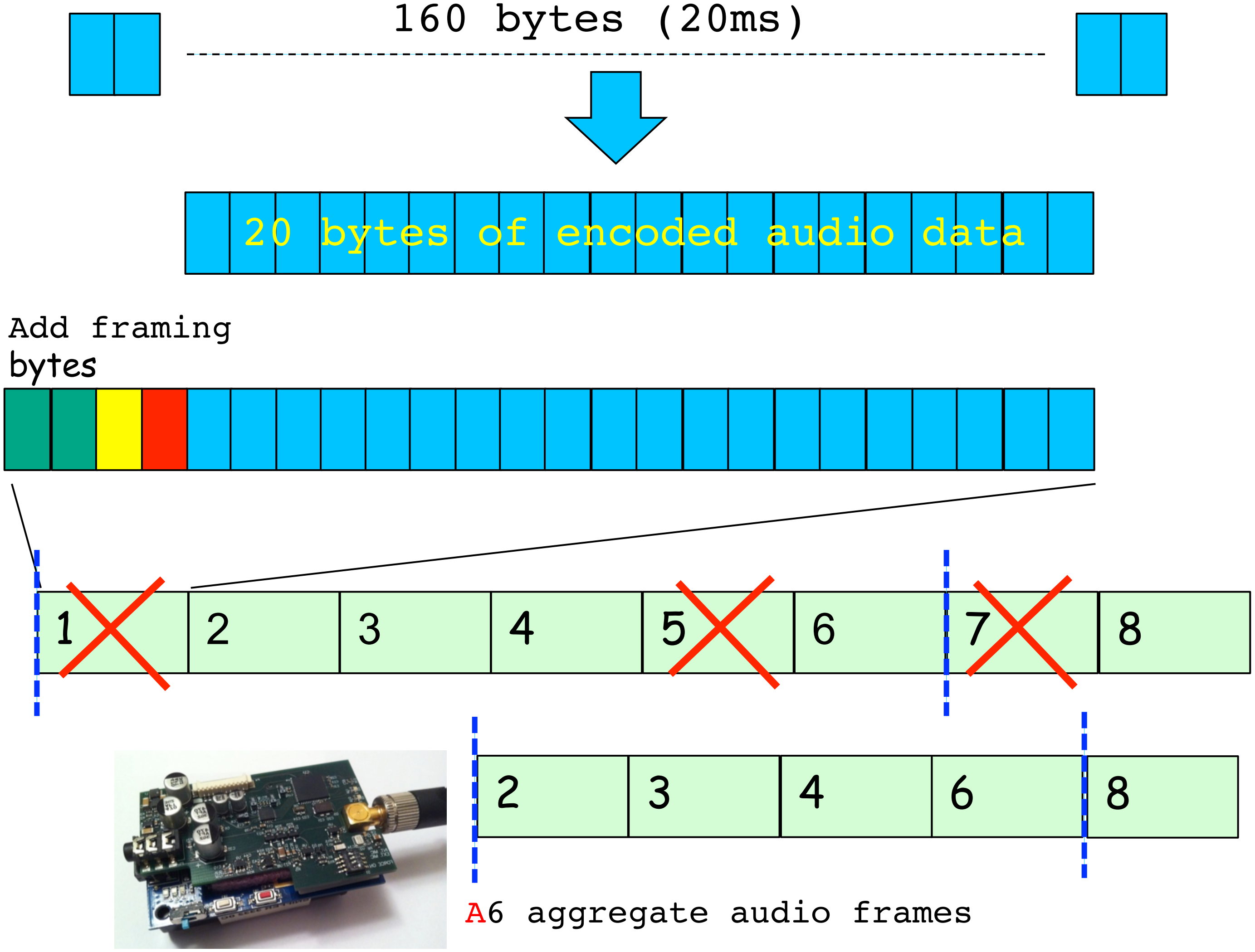


Relay node performances





speex at 8kbps on slow relay nodes



160 bytes (20ms)

20 bytes of encoded audio data

Add framing bytes

A6 aggregate audio frames

Capture 6 audio frames (120ms) but only send 4

Need to be able to relay 96-byte pkt every 120ms