



the sounds of smart environment



EAR-IT, Acoustic Sensing in Smart Environment: a case for audio streaming with low-resource IoT devices

C. Pham (LIUPPA lab, University of Pau & EGM)

Senzation'14, Biograd na Moru, Croatia. September 1-5, 2014



**MANDAT
INTERNATIONAL**



Internet of Things: Communicating Objects!

- Native communication:



- Added communication

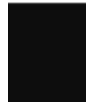
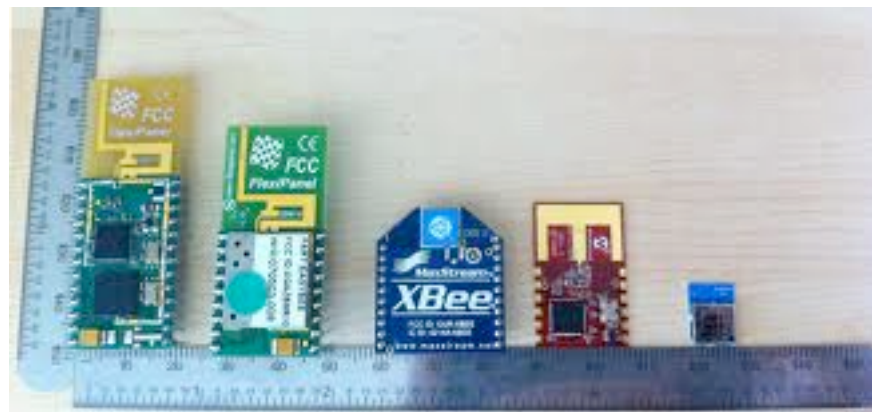
Active communication



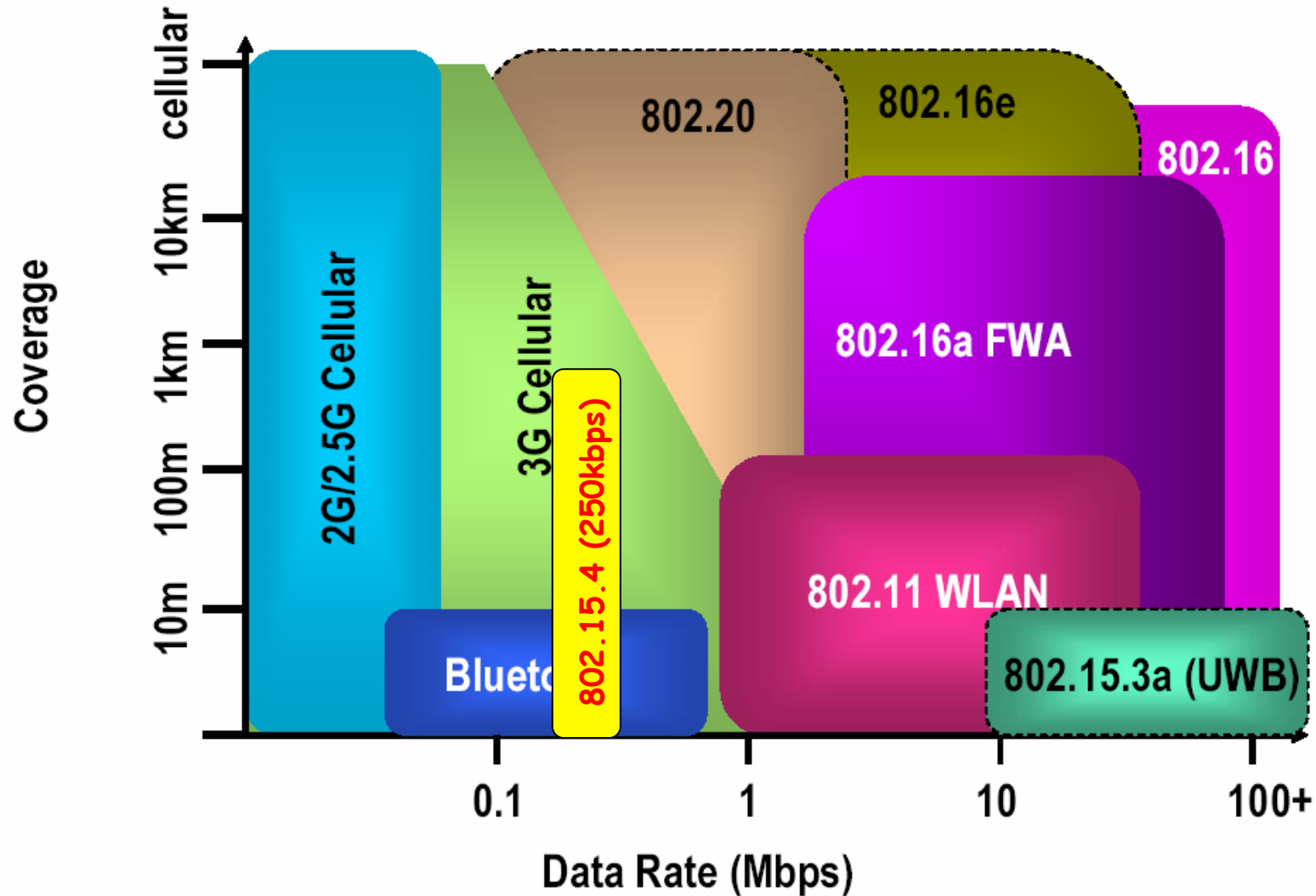
Passive communication



Wireless Communication made easy

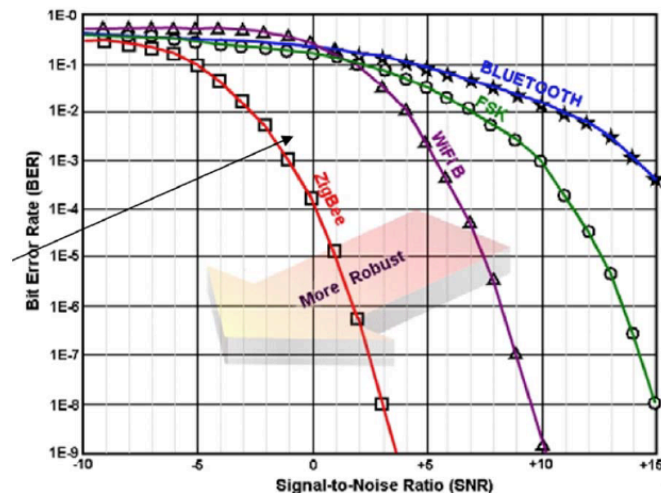


Wireless technologies



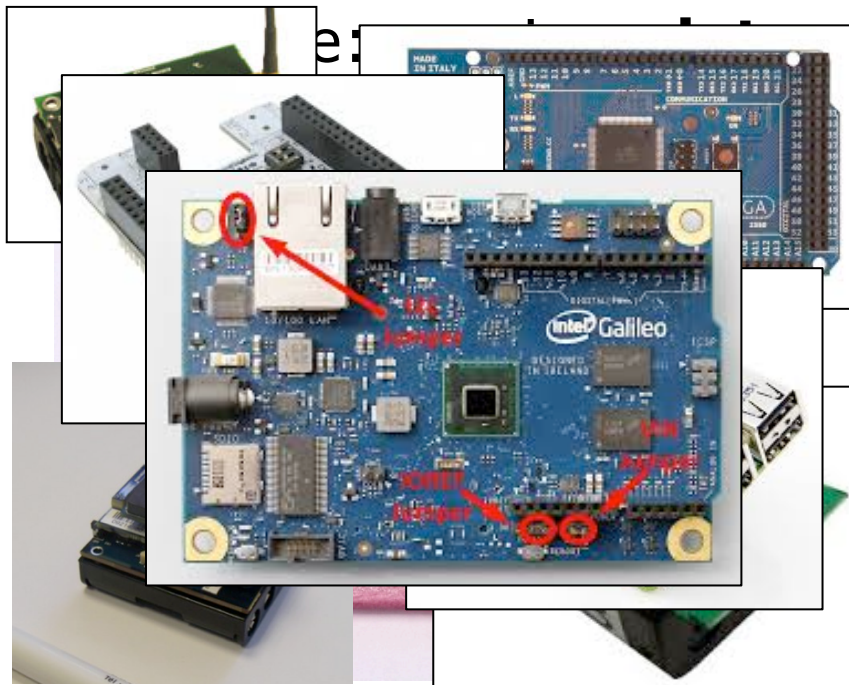
IEEE 802.15.4

- Low-power radio in the 2.4GHz band offering 250kbps throughput at physical layer
- Power transmission from 1mW to 100mW for range from 100m to about 1km is LOS
- CSMA/CA (beacon & non beacon)
- Used as physical layer in ZigBee

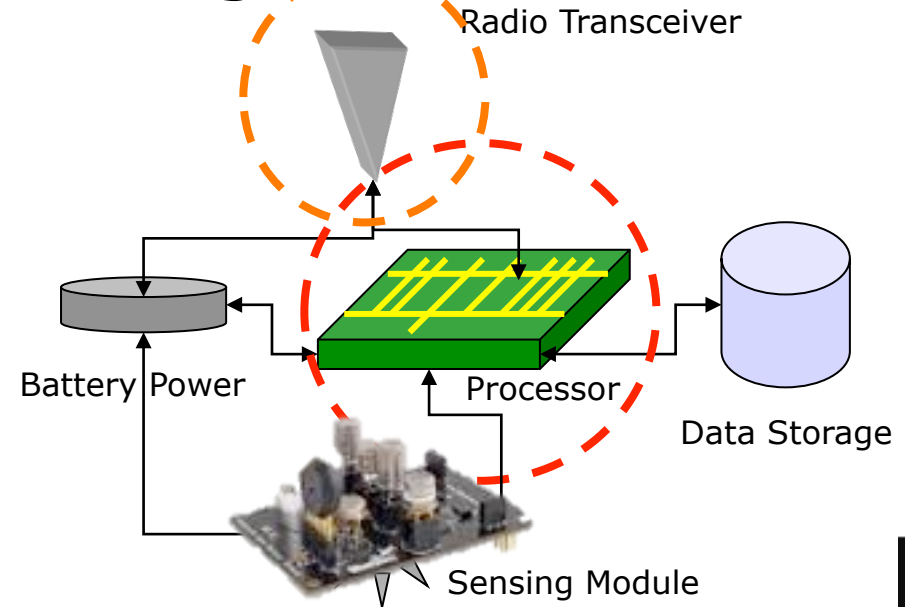


Wireless autonomous sensor

- Wireless Sensor Nodes or embedded Linux still remain the main IoT development platform
- In general: low cost, low power (the battery may not be replaceable), small size, prone to failure, possibly disposable



processing, communication





Are you I-o-T or WSN?



**IP integration, WWW
IPv6
Inter-operability
Interactions (all kind)
Semantic, Ontology
Data representation
Data logging
WebServices**



**Organization
Programmability
Energy saving
Scheduling
Efficient MAC, routing
Congestion control
Data transmission**

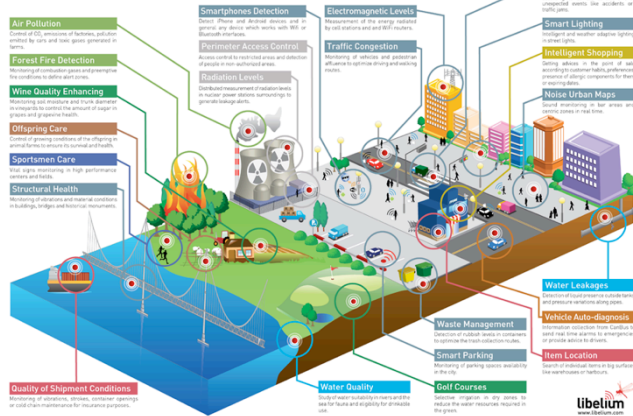


the sounds of smart environments



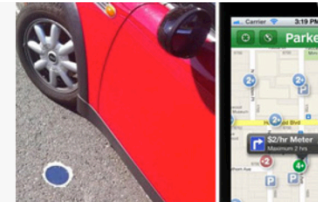
A business model in SmartCities

Libelium Smart World



KEEP STREETS CLEAN

Products like the cellular communication enabled Smart Belly trash use real-time data collection and alerts to let municipal services know when a bin needs to be emptied. This information can drastically reduce the number of pick-ups required, and translates into fuel and financial savings for communities service departments. // [Visit](#)



STOP DRIVING IN CIRCLES

With the use of installed sensors, mobile apps, and real-time web applications like those provided in Streetline's ParkSight service, cities can optimize revenue, parking space availability and enable citizens to reduce their environmental impact by helping them quickly find an open spot for their cars. // [Visit](#)



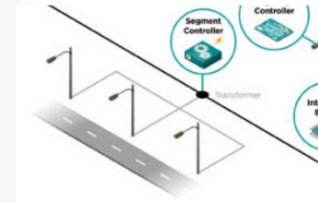
RECEIVE POLLUTION WARNINGS

The DontFlushMe project by Leif Percifield is an example that combines sensors installed in Combined Sewer Overflows (CSOs) with alerts to local residents so they can avoid polluting local waterways with raw sewage by not flushing their toilets during overflow events. // [Visit](#)



USE ELECTRICITY MORE EFFICIENTLY

The SenseNET system uses battery-powered clamp sensors to quickly measure current on a line, calculate consumption levels, and send that data to a hosted application for analysis. Significant financial and energy resources are saved as the clamps can easily identify meter tampering issues, general malfunctions, and any installation issues in the system. // [Visit](#)



LIGHT STREETS MORE EFFECTIVELY

This smart lighting system from Echelon allows a city to intelligently provide the right level of lighting needed by time of day, season, and weather conditions. Cities have shown a reduction in street lighting energy use by up to 30% using solutions like this. // [Visit](#)



SHARE YOUR FINDINGS

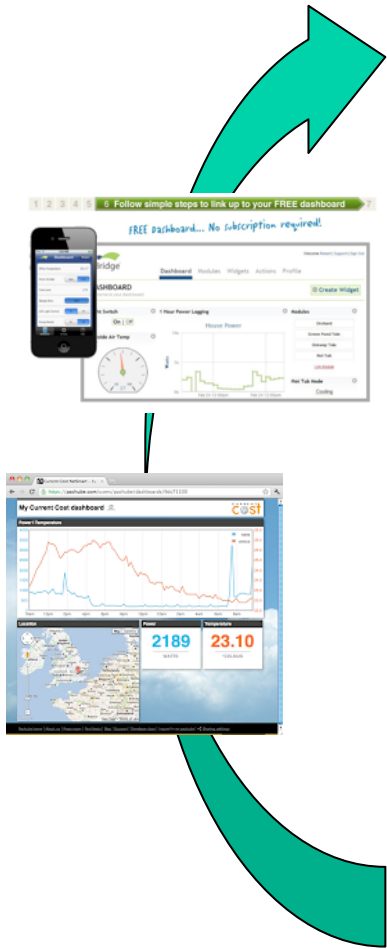
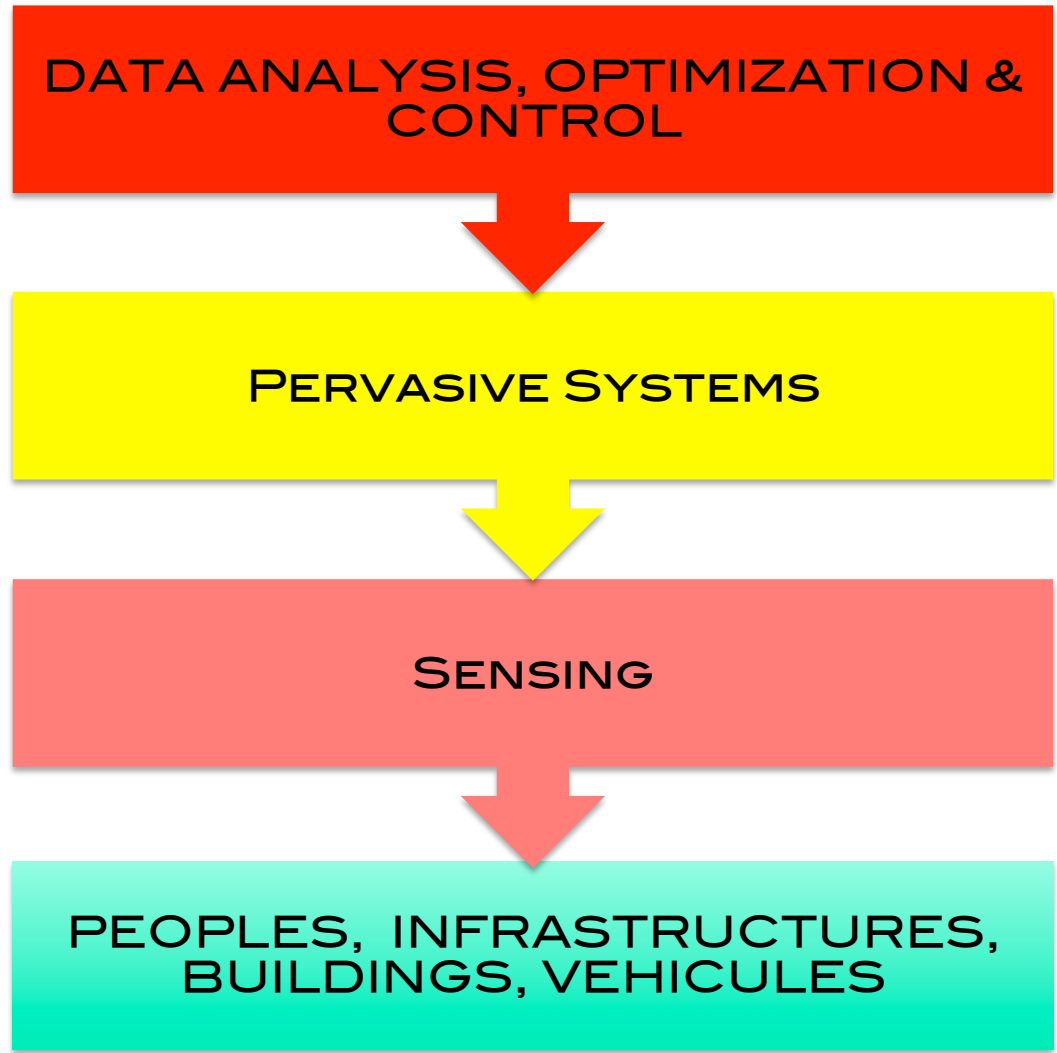
AirCasting is a platform for recording, mapping, and sharing health and environmental data using your smartphone. Each AirCasting session lets you capture real-world measurements (Sound levels recorded by their phone microphone; Temperature, humidity, carbon monoxide (CO) and nitrogen dioxide (NO₂) gas concentrations), and share it via the CrowdMap with your community. // [Visit](#)



the sounds of smart environments



Control, optimize & instrument





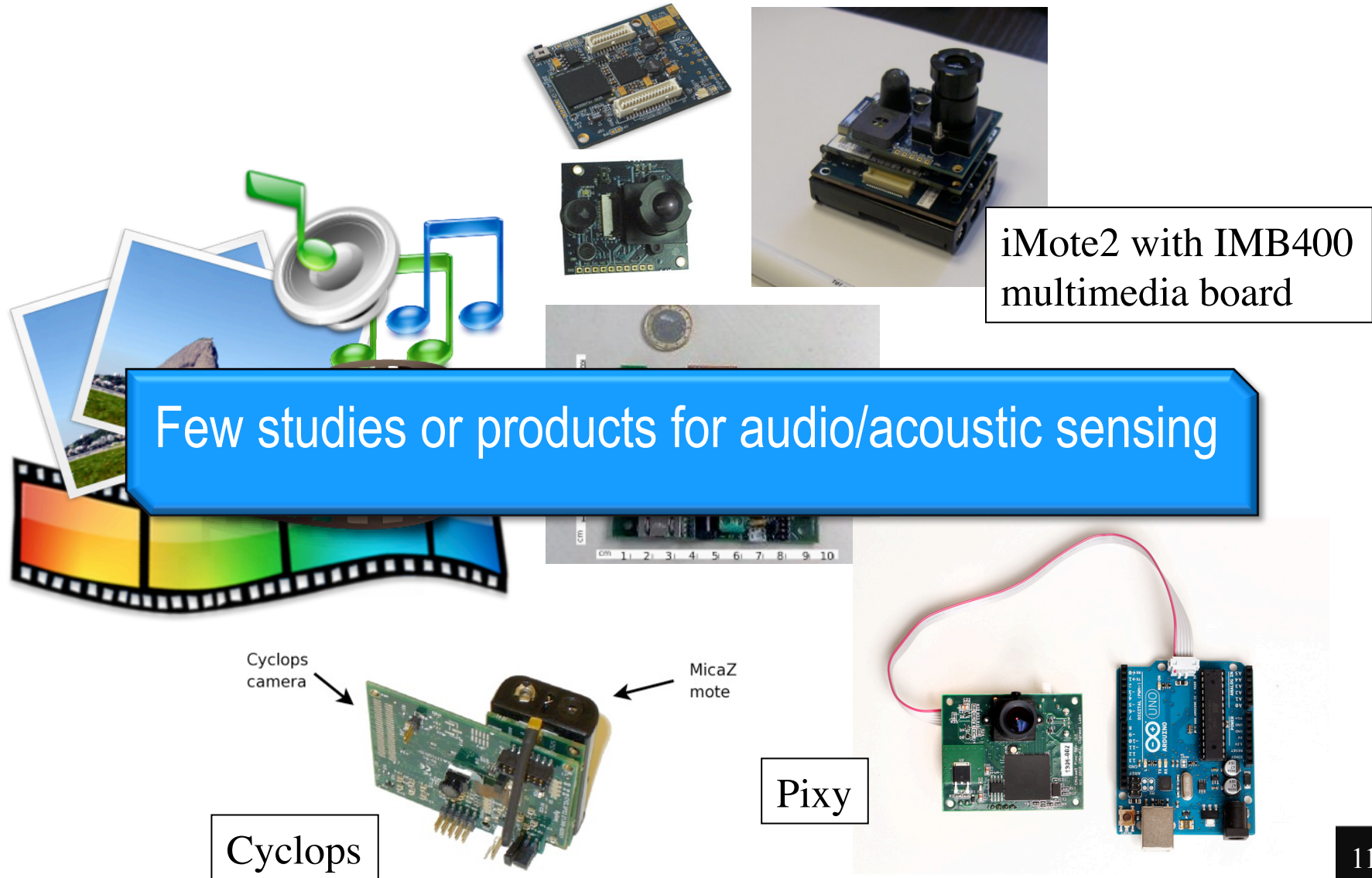
the sounds of smart environments



Towards multimedia information



Towards multimedia information



Few studies or products for audio/acoustic sensing

iMote2 with IMB400 multimedia board

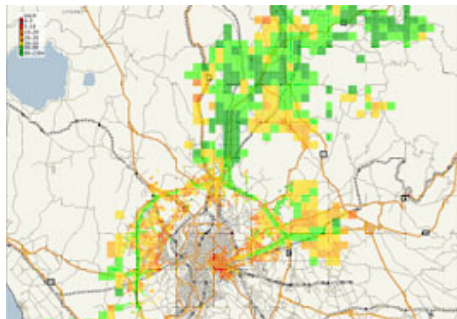
Cyclops camera

MicaZ mote

Cyclops

Pixy

Exploiting Acoustic data



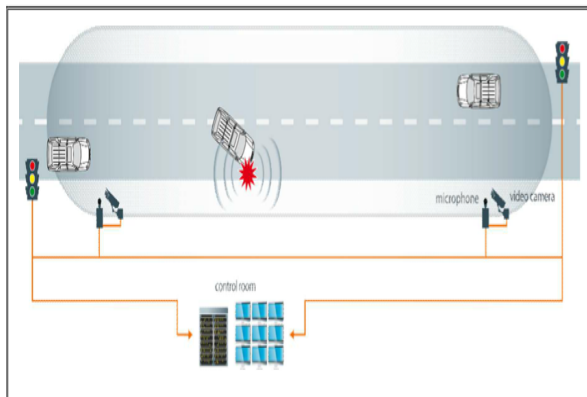
Sound Mappings



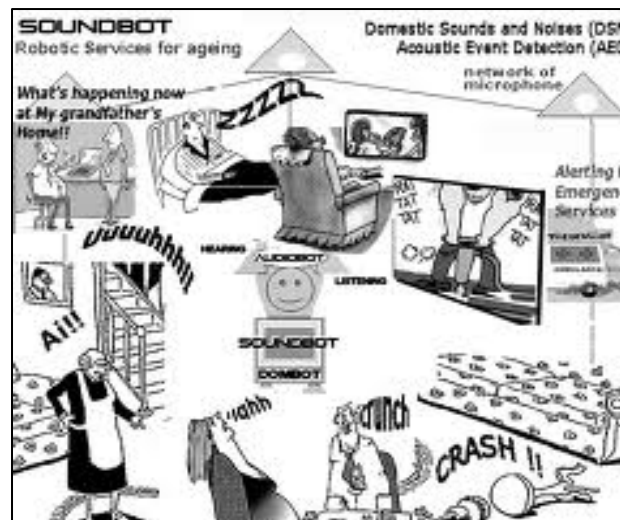
Traffic Management



Energy-efficiency



Traffic Accidents



Ambient Assisted Living



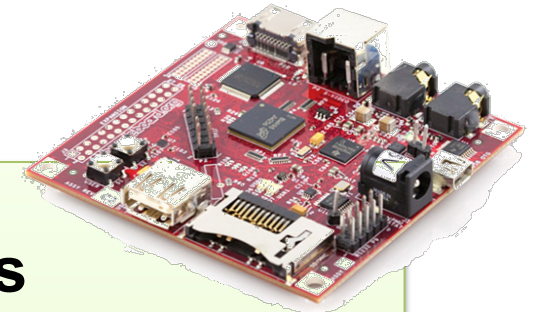
Use of Complementary Technologies



Today's audio-ready IoT Technologies

- *Less Capable*
- *Low energy*
- *Cheaper to get*
- *Much deployed*

www.ear-it.eu



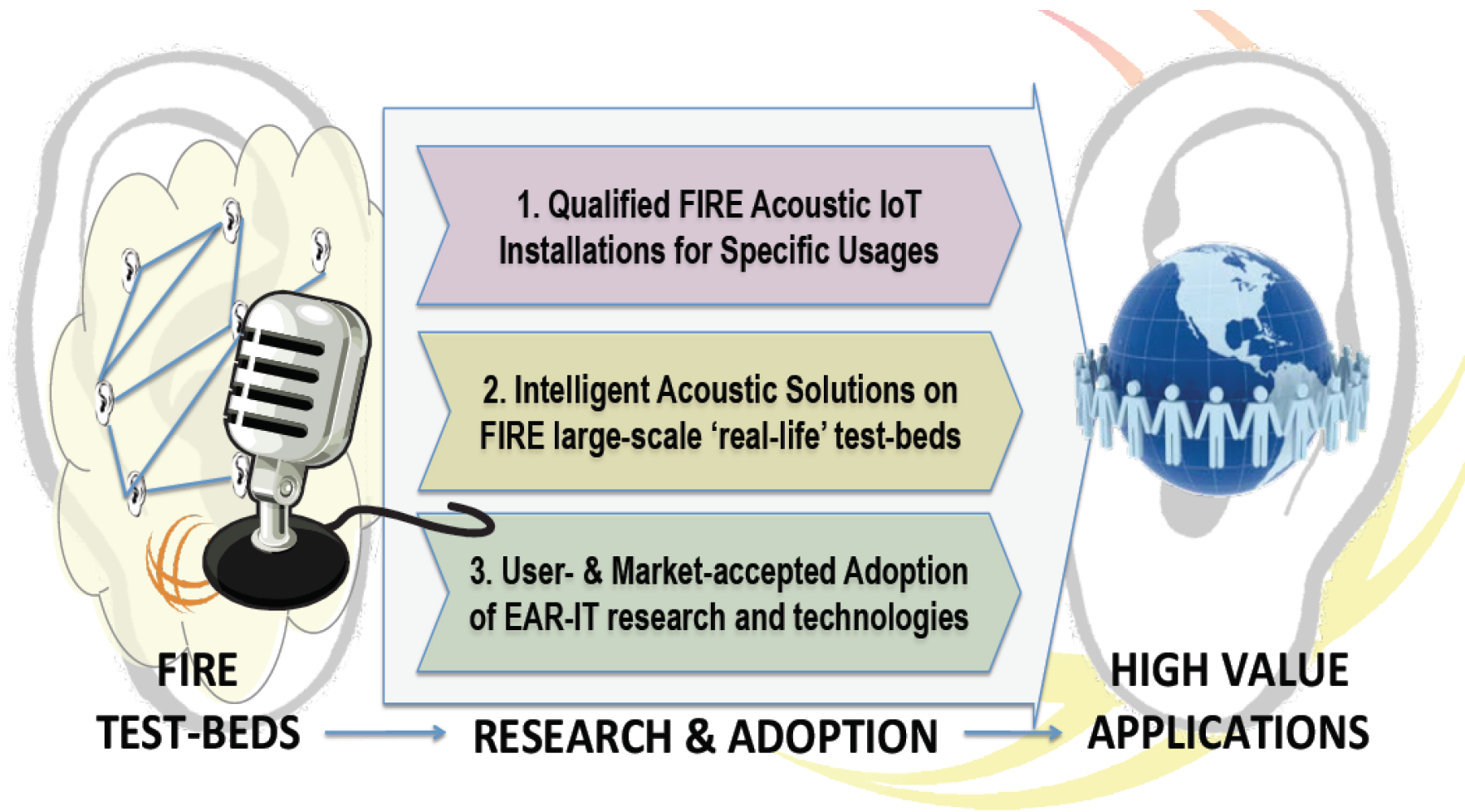
Today's Acoustic Sensing Technologies

- *Powerful*
- *Power greedy*
- *More costly*
- *Fewer available*



Combining the complementary Acoustic Sensing and Internet-of-Things technologies of today for value

EAR-IT research experiment approach




EAR-IT test-beds



Smart Santander
<http://www.smartsantander.eu/>



HOBNET
www.hobnet-project.eu



Experimenting Acoustics in Real environment using Innovative Test-beds

SmartSantander test-bed



SmartSantander aims at providing a European **experimental test facility** for the **research** and **experimentation** of architectures, key enabling technologies, **services** and applications for the Internet of Things (IoT) in the context of the **smart city**.



Smart Santander Highlights

- ❑ **Targeting:**
 - Researchers
 - End users
 - Service providers
- ❑ **Duration**
 - 36 months
- ❑ **Consortium**
 - 15 Organisations
 - 8 EU countries + AU
- ❑ **Budget / Funding**
 - 8.6 M€ / 6 M€
- ❑ **Resources**
 - 746.2 PM



the sounds of smart environments

SmartSantander test-bed

Santander's sensor network deployment



SmartSantander test-bed

Santander's sensor network deployment




SmartSantander IoT node

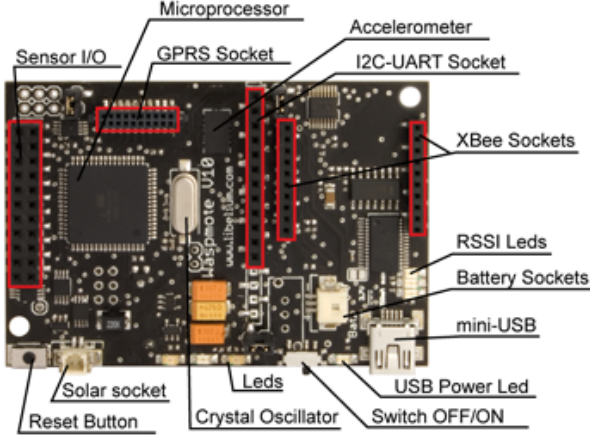


ATmega1281 microcontroller
 8Mhz, 4K RAM & 2G SD card.
 2.4GHz IEEE 802.15.4 XBee
 Libelium API v031

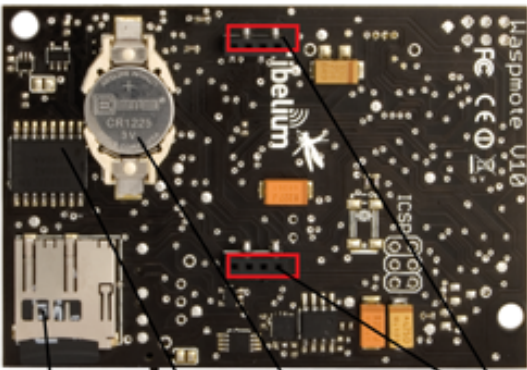
Gases



- 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




Labels: Microprocessor, Accelerometer, Sensor I/O, GPRS Socket, I2C-UART Socket, Xbee Sockets, RSSI Leds, Battery Sockets, mini-USB, Solar socket, Leds, USB Power Led, Reset Button, Crystal Oscillator, Switch OFF/ON



Labels: SD CARD, RTC, Aux. Battery, GPS Socket

Events

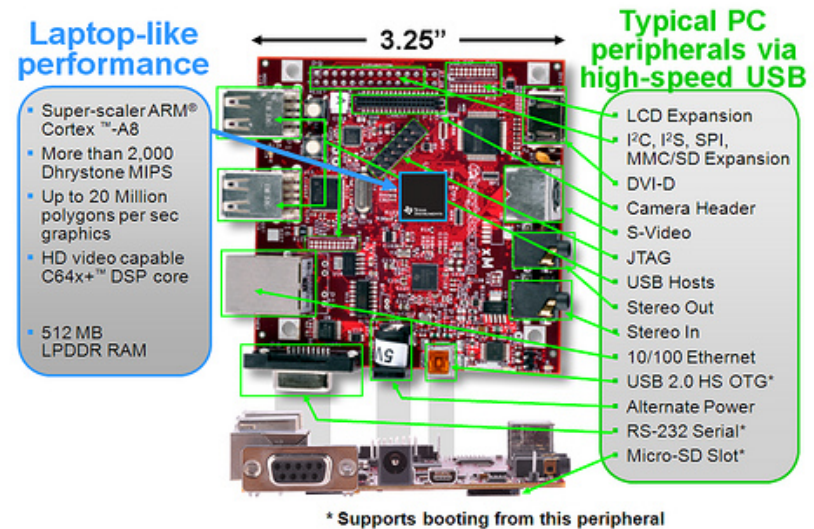
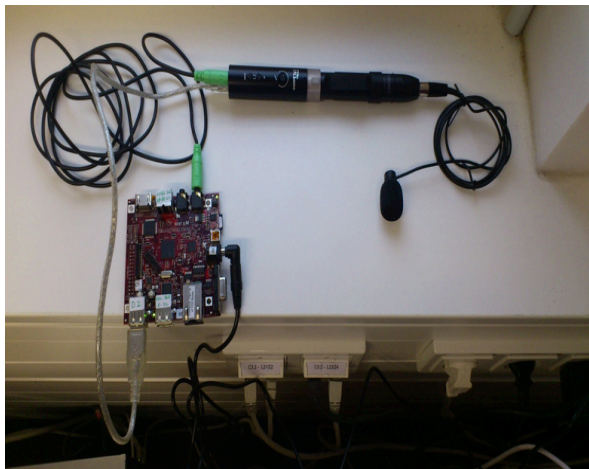


- Pressure/Weight
- Bend
- Vibration
- Impact
- Hall Effect
- Tilt
- Temperature (+/-)
- Liquid Presence
- Liquid Level
- Luminosity
- Presence (PIR)
- Stretch

Adding Acoustic Processing Units

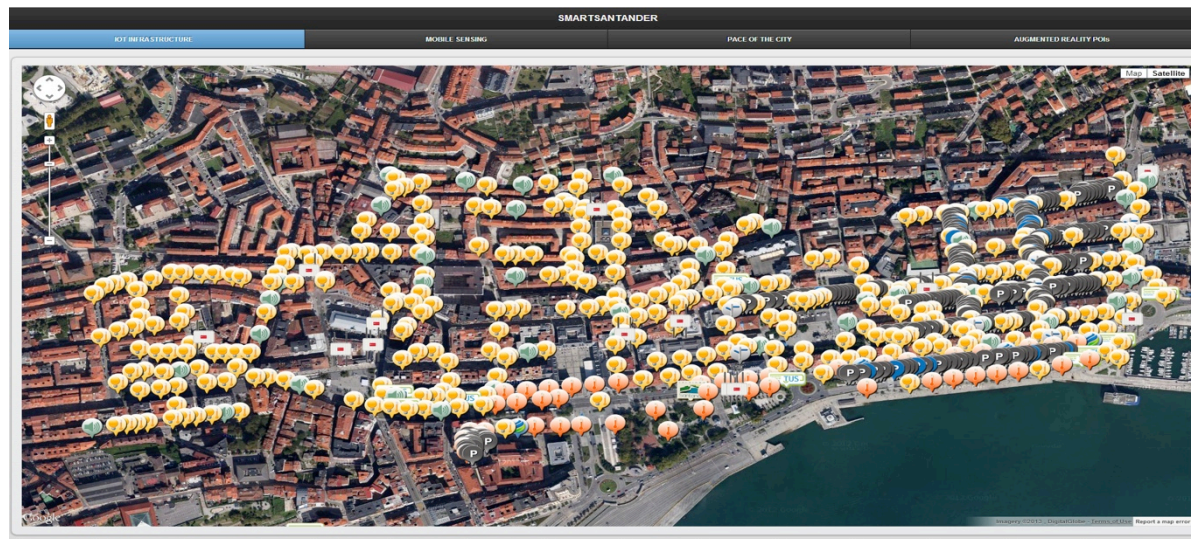
EAR-IT's APUs – Acoustic Processing Units

- Computer-like performance processing platform able to run robust accosting sensing framework;
- “Soundboard” - high quality sound, customized and stackable sound card;
- High-quality & responsive microphones;



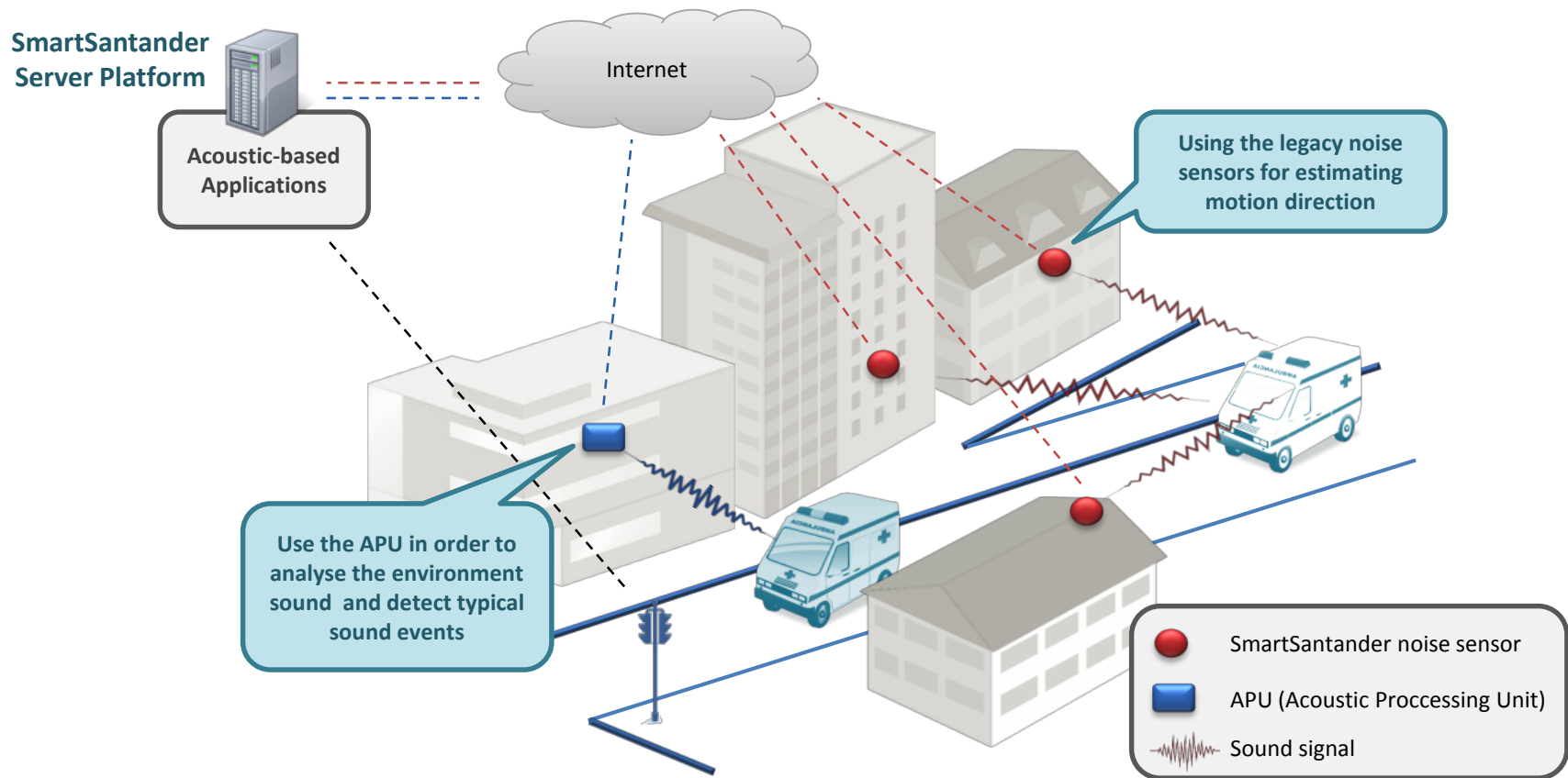
SmartSantander meets EAR-IT

Upon concrete noise pattern detected by the APU, legacy sensors collect data for several purposes \Rightarrow Two use cases as a starting point.



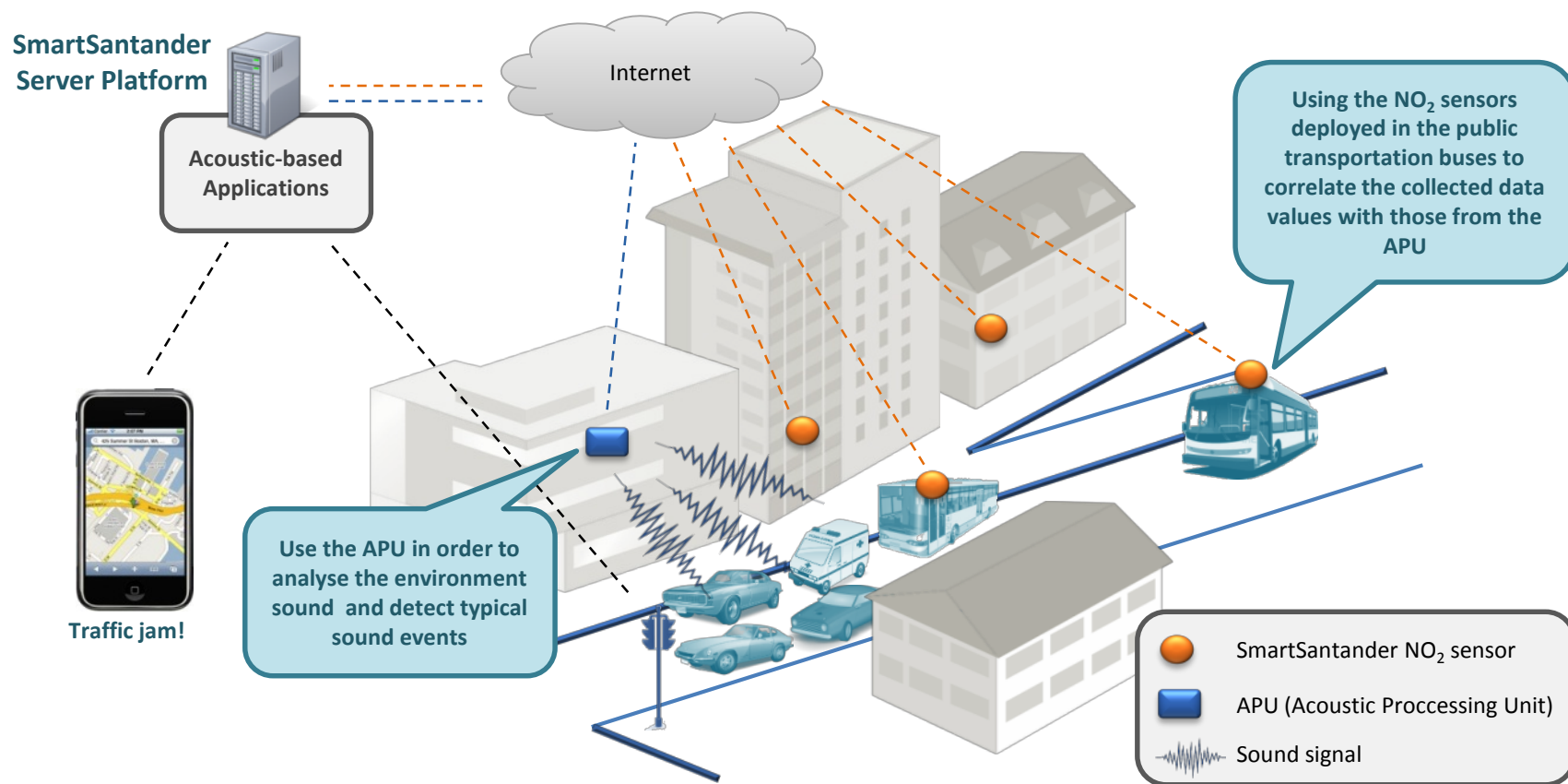
EAR-IT Use Case 1: Emergency Detection

Use the APU to detect an alarm \Rightarrow Legacy SmartSantander noise sensors to get the direction of such an event (police car, ambulance,...).



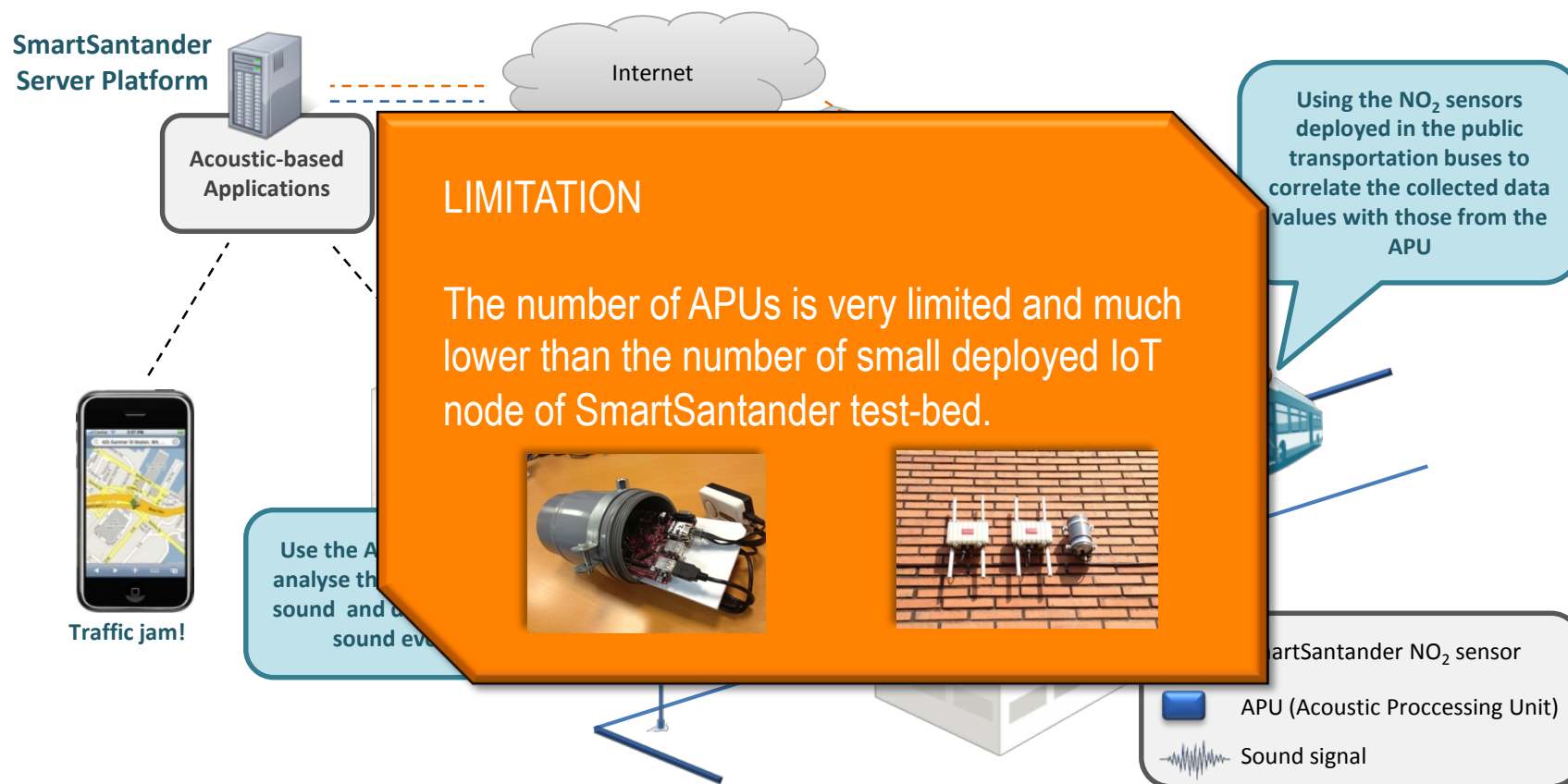
EAR-IT Use Case 2 : Traffic Monitoring

Use the APU to measure the traffic density and correlating it with pollution values (NO₂, CO,...) collected by legacy fixed and mobile nodes in the area.

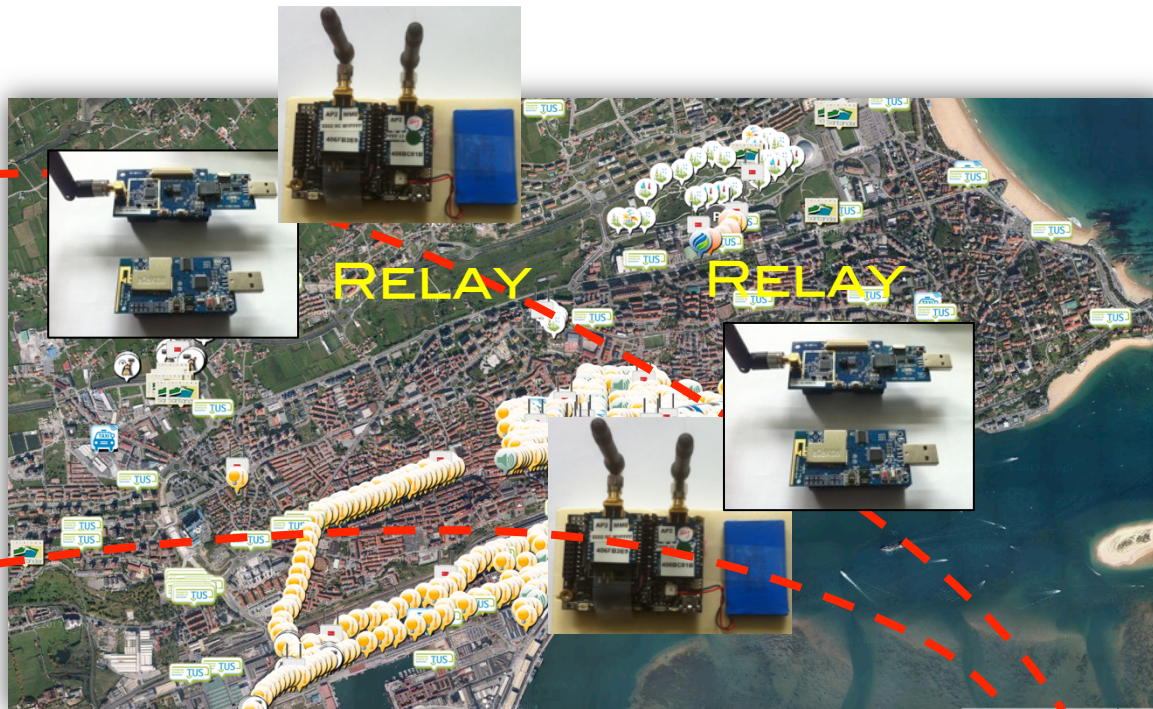


EAR-IT Use Case 2 : Traffic Monitoring

Use the APU to measure the traffic density and correlating it with pollution values (NO₂, CO,...) collected by legacy fixed and mobile nodes in the area.



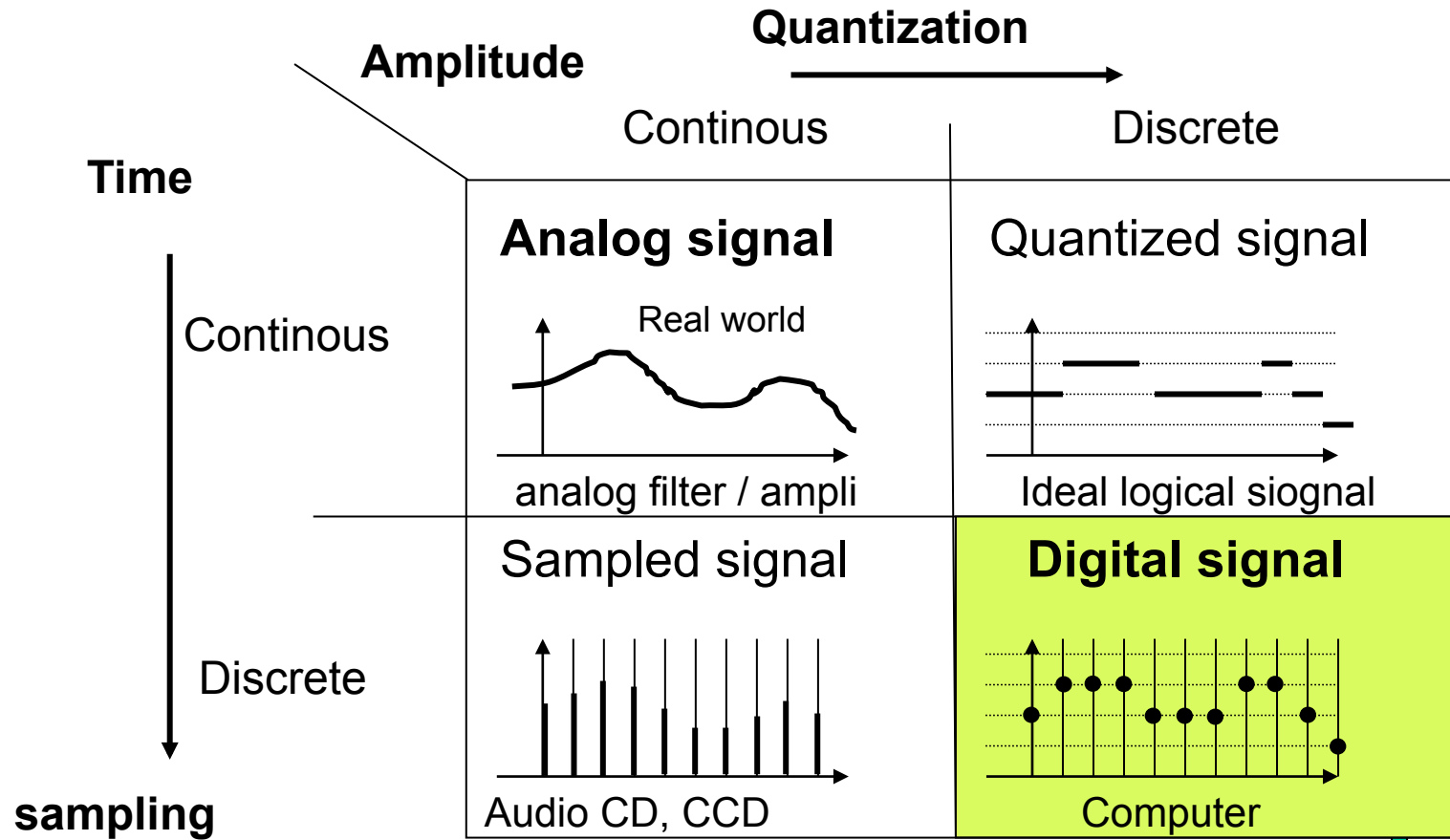
Use deployed low-resource IoT node to enhance acoustic services



PLAY/STORE RECEIVED
AUDIO DATA



Review of digital audio



Only in this case can we associate an integer value to the signal



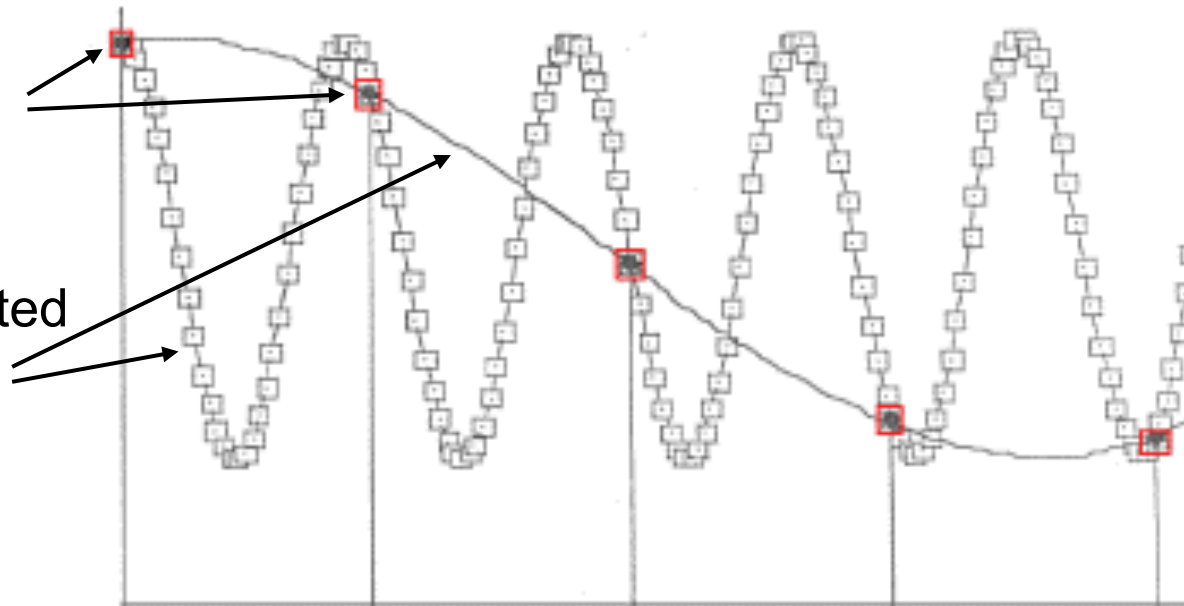
Sampling: Shannon's theorem

Shannon's theorem: $f_s > 2 \times f_{max}(\text{Signal})$

Example :

Samples

Extrapolated
signal ?



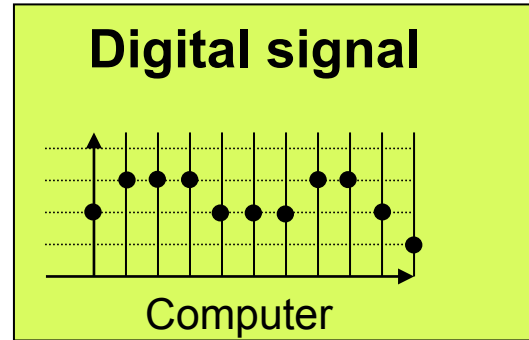
An incorrectly sampled signal will not be reconstituted



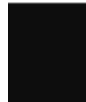
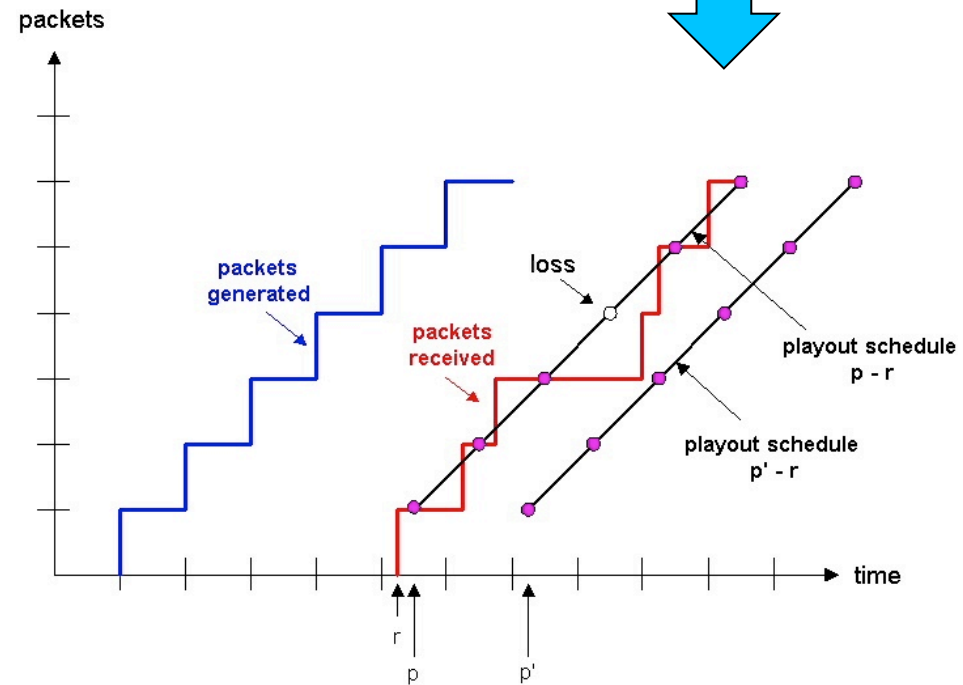
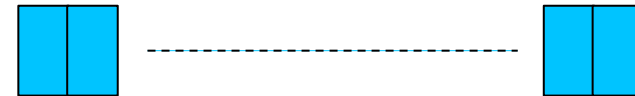
Narrow-band audio

- Sampling rate up to 8kHz
- 1 sample every $1/8000$ s (125 μ s)
- Sample coded on 8 bits
- Raw throughput of 64kbps
- So-called Pulse Code Modulation (PCM) used in most wired telephony systems
- With 4kHz sampling rate, can reduce to 32kbps

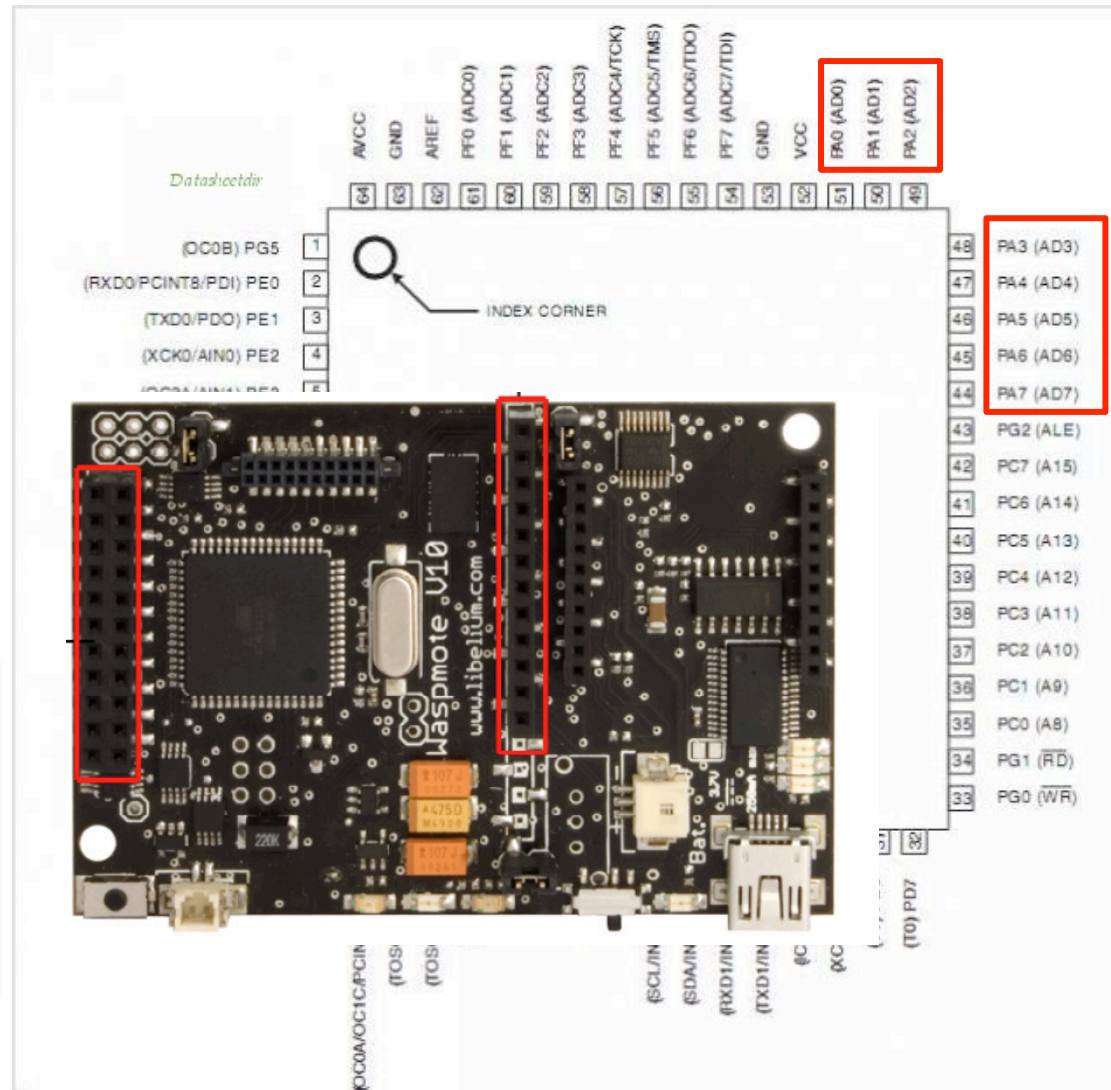
Audio streaming principle



160 8-bit samples (20ms)



ucontroller vs microprocessor



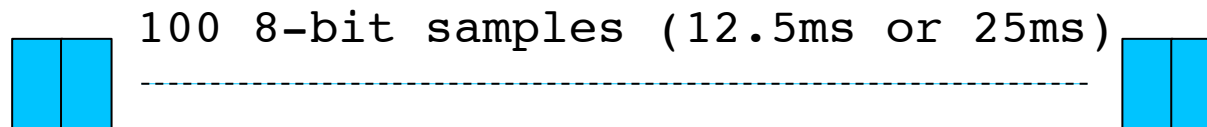
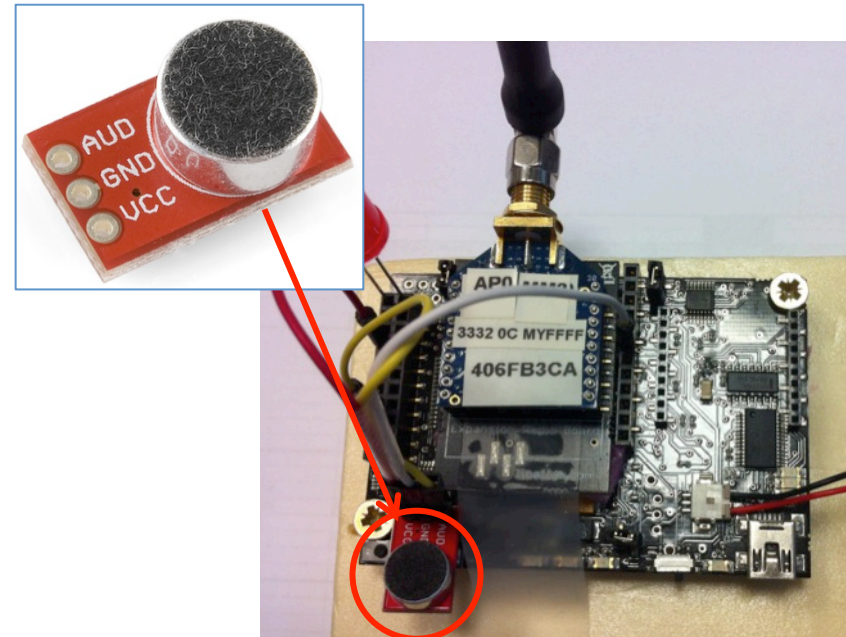
Input voltage between 0 and Vref (e.g. 3.3V). ADC usually have 10-bit resolution:

0 is for 0V
1014 is for 3.3V

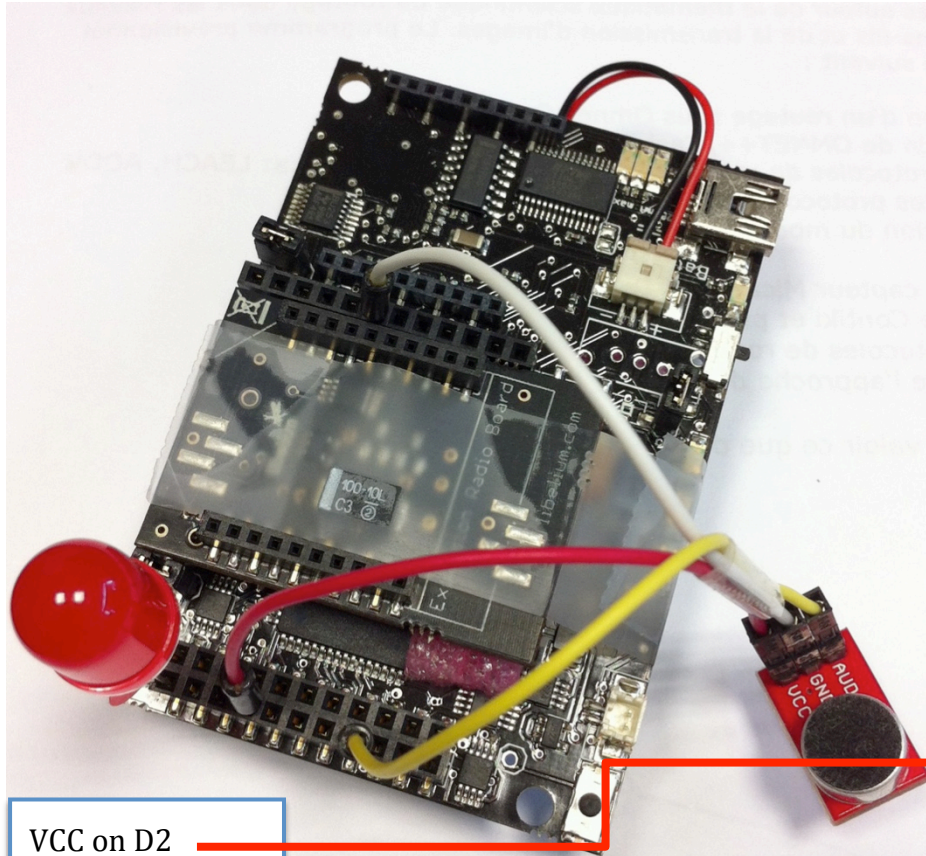


Practical audio

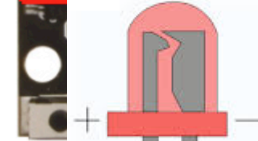
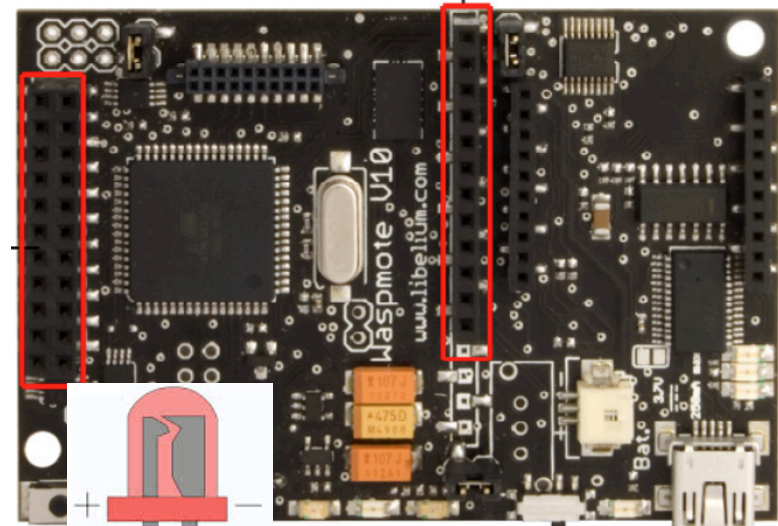
- Electret mic with amplifier on ADC input pin
- Convert from 10-bit to 8-bit sample
- 8Khz sampling gives 64000bps
- 4Khz sampling gives 32000bps



Details of pin connection



VCC on D2
 AUDIO on A2
 GND on GND



DIGITAL0	GND	AUX-SERIAL-1-TX
DIGITAL6	DIGITAL7	AUX-SERIAL-1-RX
DIGITAL4	DIGITAL5	AUX-SERIAL-2-RX
DIGITAL2	DIGITAL3	AUX-SERIAL-2-TX
RESERVED	DIGITAL1	RESERVED
ANALOG6	ANALOG7	GND
ANALOG4	ANALOG5	GND
ANALOG2	ANALOG3	MUX_RX
SENSOR POWER	ANALOG1	MUX_TX
GPS POWER	5V SENSOR POWER	SENSOR POWER
SDA	SCL	SCL
		SDA

Simple program

```

#define TIMING_SAMPLING 125 // 8000Hz
#define CAPTURE_DURATION 15000000UL // in us 15s
#define SAMPLE_COUNT_CAPTURE CAPTURE_DURATION/TIMING_SAMPLING

void setup() {
    Timer1.initialize(TIMING_SAMPLING);
}

void callback () {

    sampleCount++;

    if (sampleCount < SAMPLE_COUNT_CAPTURE) {

        val = analogRead(ANALOG2) ; // read analog value
        val8bit = (val >> 2) ) ; // convert into 8 bit

        // write on UART1
        serialWrite(val8bit,1);
    } else {

        stopCapture=true;

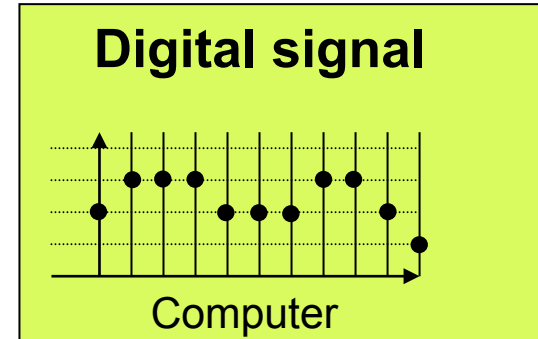
        Timer1.detachInterrupt();
    }
}

void loop() {

    // we have to go to sleep
    if ( (millis() - lastWakeupTime > 15000 || stopCapture) && capturingAudio) {
        capturingAudio = false;
        stopCapture = false;
    }

    // we have to wake up
    if (millis() - lastSleepTime > 15000 && !capturingAudio) {
        sampleCount=0L;
        lastWakeupTime = millis();
        capturingAudio = true;
        Timer1.attachInterrupt(callback);
    }
}

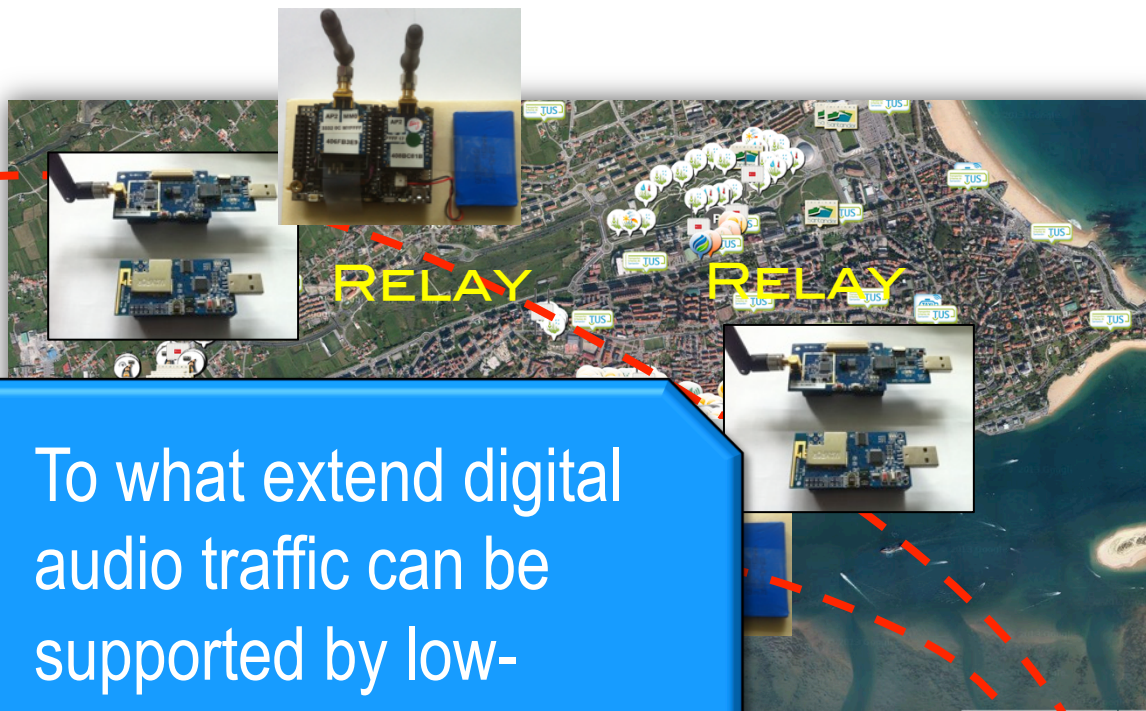
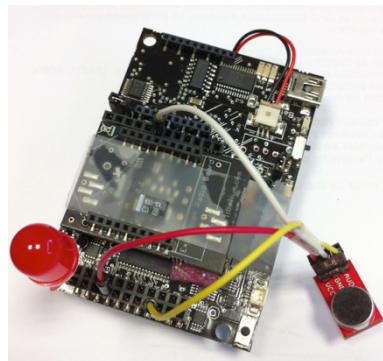
```



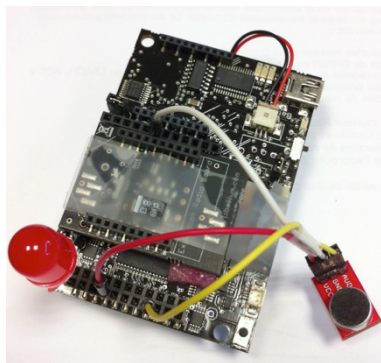
Get digital value of sound pressure level, i.e. raw audio

Or any other way to put the sample into a transmission buffer

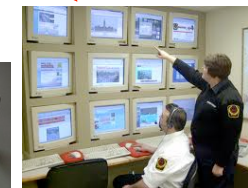
Use deployed low-resource IoT node to enhance acoustic services



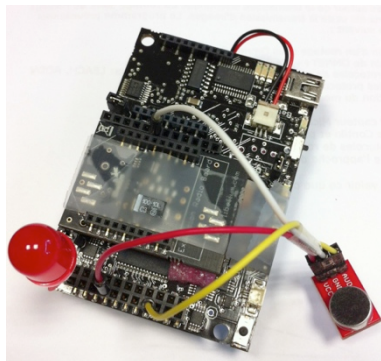
To what extent digital audio traffic can be supported by low-resource IoT nodes?



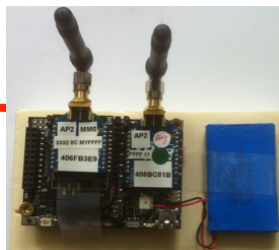
PLAY/STORE RECEIVED AUDIO DATA



Multi-hop audio streaming

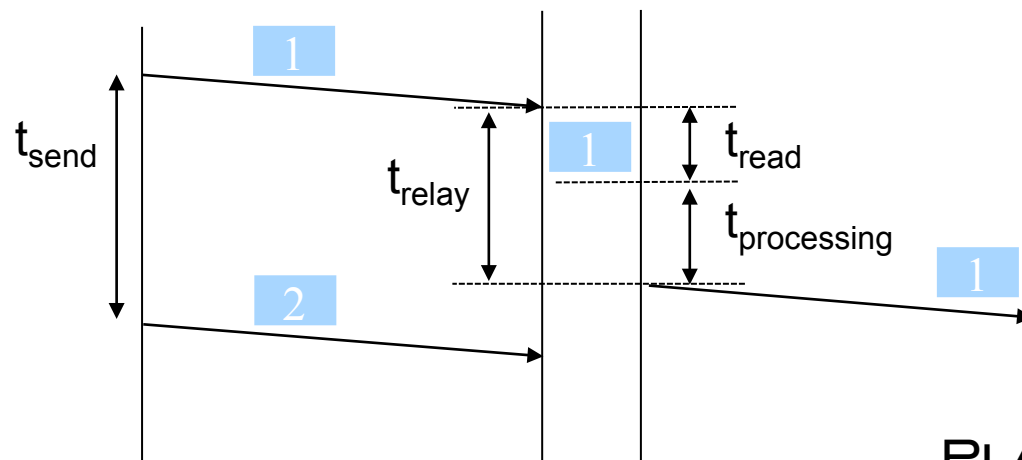


SENDS AUDIO DATA



RELAY

RELAY



PLAY RECEIVED FILE





Communication performances

- Application level performances depends on OS, API, hardware architecture
- Usually much lower than radio performances
- What are minimum latencies & max. throughput?
 - For sending?
 - For receiving?
 - For relaying?



Mass-market sensors

8MHz Atmega1281
8kB SRAM, 128kB Flash
Xbee radio

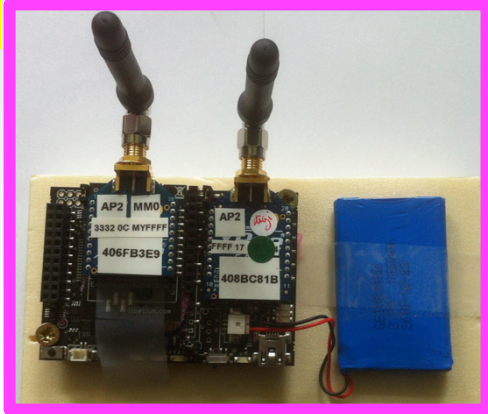


waspmote

**COST:
~100€**

LIBELIUM WASPMOTE

The image shows a Wasp mote sensor board, which is a small, compact device with a black antenna and a blue PCB. It is shown in two views: a top-down view and a side view. The side view shows the board's components, including a blue PCB, a black antenna, and a blue battery. The top-down view shows the board's components, including a black antenna, a blue PCB, and a blue battery.



**COST:
~80€**



ARDUINO MEGA2560

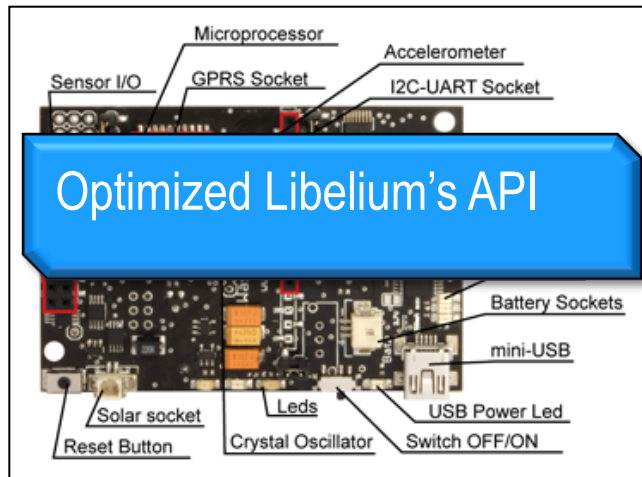
16MHz Atmega1281
8kB SRAM, 128kB Flash
Xbee radio





Sensor's HW&SW

LIBELIUM WASPMOTE



Optimized Libelium's API



A. Rapp's XBee lib & API

ARDUINO MEGA2560

UART-based connection to micro-controller

Default speed is usually 38400 bauds

Higher baud rate are possible but...

```

WaspXBee802_2_traffic_generator | Waspnote-IDE 02
-----Waspnote XBee 802.15.4 Traffic Generator -----
Version:      0.33
Design:      C. Pham
Implementation: C. Pham

"Z100"      : set packet size to 20 bytes
"Z200"      : increases pkt size from 5 bytes to 100 bytes (or 400 bytes with Libelium API) every 20pkt
"Z300"      : set frequency to 194.7/200mhz
"Z0013A200400BC01F" : set destination address to 0013A200400BC01F, broadcast by default 000000000000FFFF
"ALF"/"ADM" : enable/disable Libelium API with WaspMote as node ID
"PL0"/"P00" : enable/disable print sent data

Jun, 14th, 2013, v0.33
adds command string prefix to "/0". All existing command should be prefixed such as: "/0Z100"
March, 19th, 2013, v.032
adds support for unsigned long time, fixes wrap around inter-packet time, adds beacon print for long inter-packet time
March, 1st, 2013, v.031a
adds support SmartSantander test-bed
Feb, 15th, 2013, v.031
adds support for periodic size increase feature
Jan, 15th, 2013, v.030
adds support for digimesh module, enable this with USE_UART1, RCV_CMD_UART0, RCV_CMD_UART1. This 2nd XBee module
adds reception cmd with Digimesh radio module, enable this with USE_UART1, RCV_CMD_UART0, RCV_CMD_DIGIMESH
adds IPRTI terminate for sending packets, en=1= with IPKE_UART1. Can force reception of commands on UART0, enable with
adds: 0: send to, or 0: send with LED
adds: 0: send to, or 0: send with LED
and GPS support
Dec, 21st, 2012, v0.2
improves version 0.1 with better timing features and statistics

/*
 * TODO
 * basic LED and GPS support need more debugging
 */

// BEGIN of compilation #define statements
//
// uses advanced timing of the Libelium send API. CAUTION: need modified version of the API
#define SEND_API_TIMING
    
```

ARDUINO-BASED IDE WITH C++-LIKE

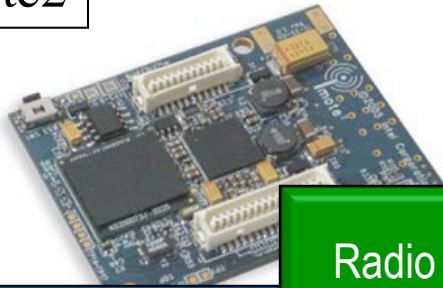


XBEE 802.15.4

« Academic » sensors



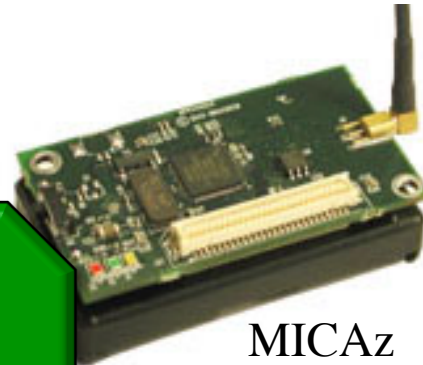
iMote2



13-416MHz PXA270
Wireless MMX DSP
256kB SRAM, 32MB
32MB SDRAM
CC2420 radio

Radio module
CC2420 is
connected
through SPI bus

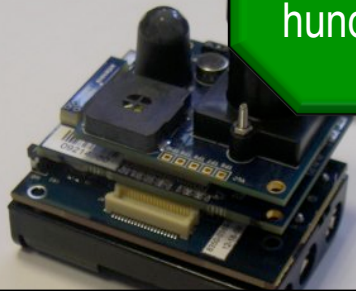
SPI speed is in
the order of
several
hundredth kbps



MICAz

8MHz Atmega128L
4kB SRAM, 128kB Flash
CC2420 radio

Motes are programmed under the
TinyOS operating system & lib



iMote2 with IMB400
multimedia board



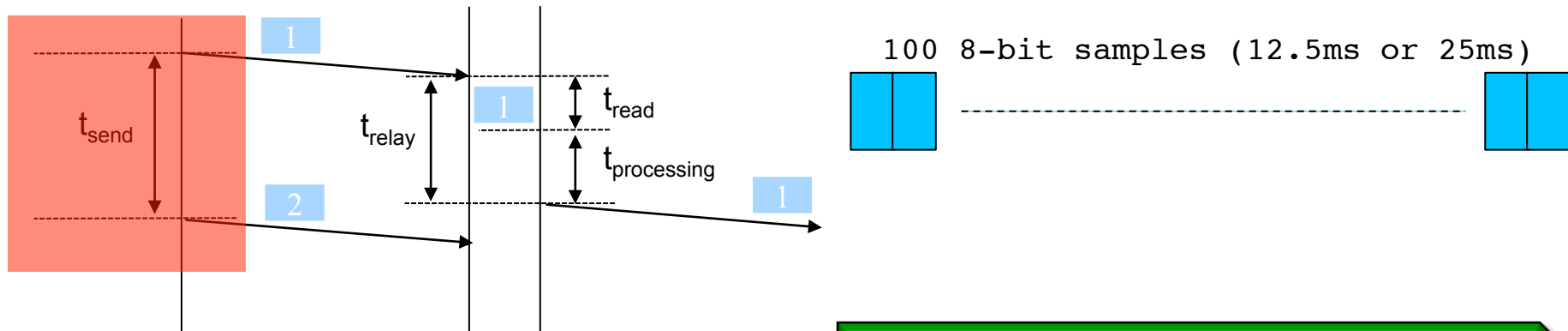
TelosB

Advanticsys CM5000 & CM3000
TelosB-like mote

8Mhz MSP430F1611
10K SRAM, 48K flash
CC2420 radio



Sending performances



TRAFFIC GENERATOR

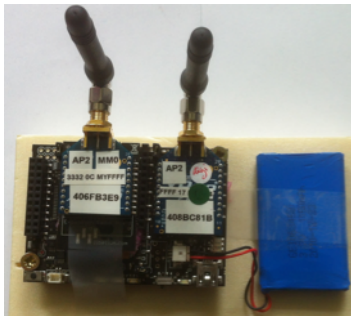
```
void loop() {
    T0;
    L0=T0;
    ...
    T1;
    send(buf);
    T2;
    ...
}
```

« Time in send() » is T2-T1
 « Time between 2 pkt generation » is T0-L0
 Time resolution is millisecond
 Minimum data manipulation

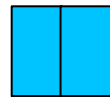
Measure the time in various part of API send () when possible.



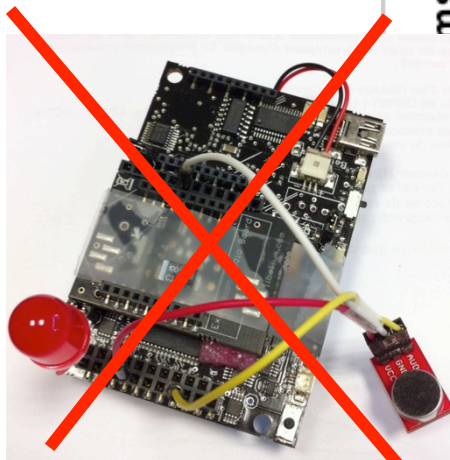
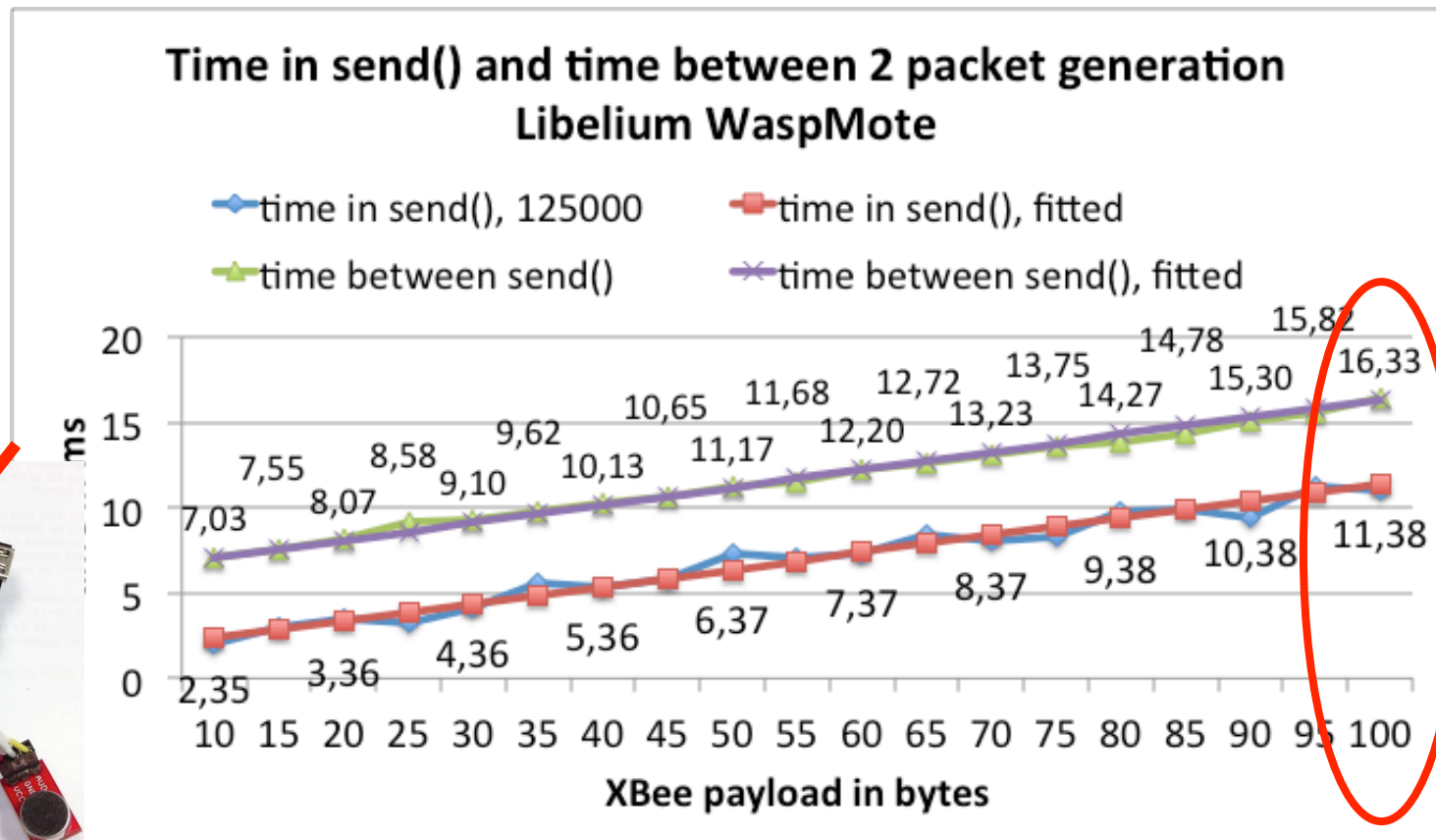
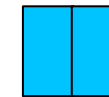
IoT node sending performance



LIBELIUM WASPMOTE

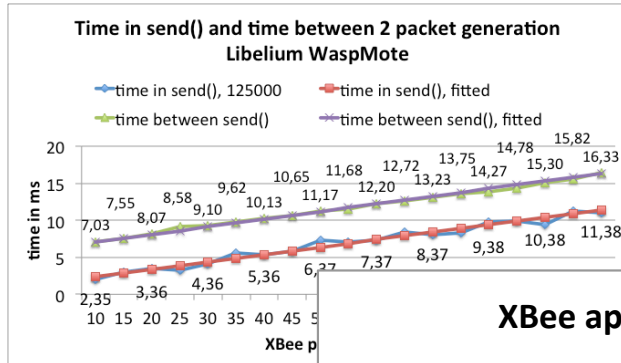


100 8-bit samples (12.5ms or 25ms)

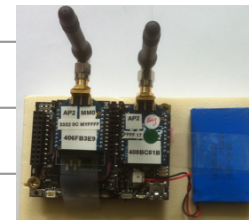
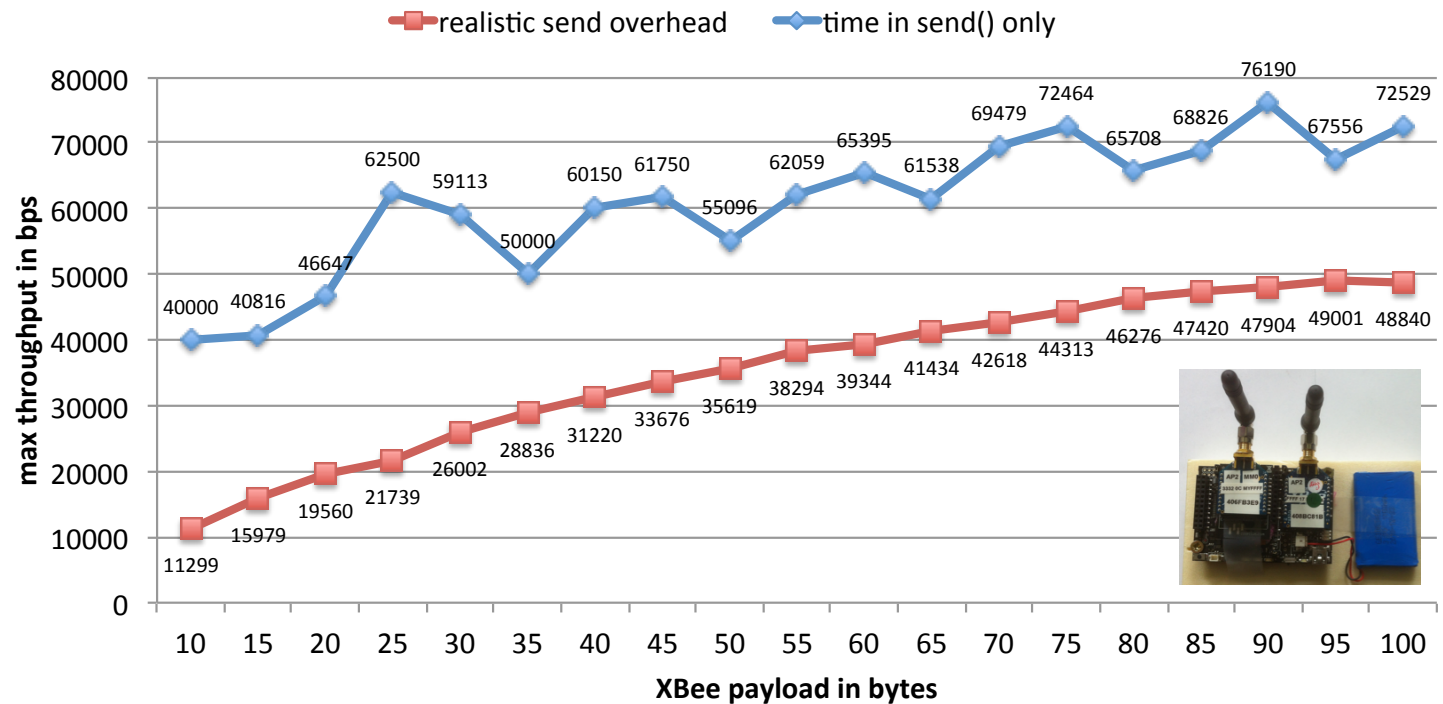


No capture and transmission at the same time if using only mote ucontroller!

Maximum app. Level throughput



XBee application level max sending throughput & realistic send overhead, WaspMote



First solution: use XBee ucontroller

- XBee 802.15.4 radio module has an embedded ucontroller that can perform framing tasks
- So called transparent mode or « serial line replacement » mode (AP0)
- Application provides payload, e.g. « hello », and XBee fills-in framing information
- Limitation is no dynamic destination address



XBee and XBee gateways

IEEE 802.15.4 MAC frame format

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



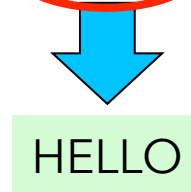
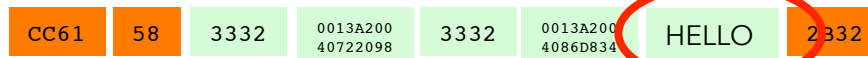
HELLO

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

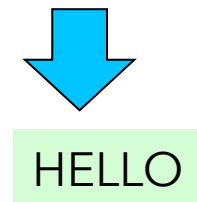
View as a serial port
/dev/ttyUSB0



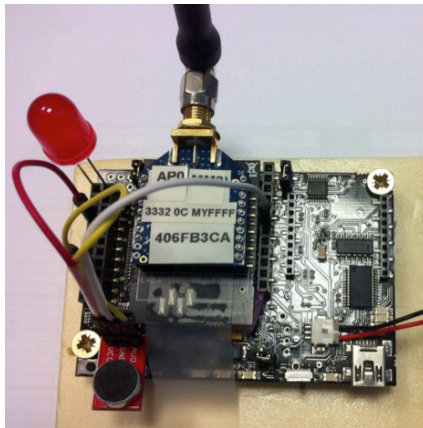
Some hardware give access to Link-layer information



Transparent mode
Or Serial line replacement mode



Example with XBee & 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 radio in AP0 mode



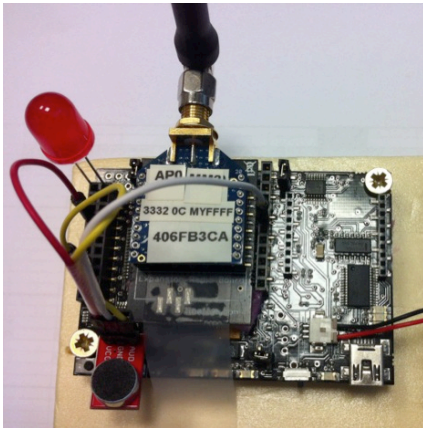
With XBee GW also in AP0 mode

Use python script to read serial port /dev/ttyUSB0

```
> python SerialToStdout | play --buffer 50 -t raw -r 8000 -u -1 -
```



Example with XBee & Xbee gw AP0



XBee radio in 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
```

Can support up to 8kHz raw audio, but:

- 1/ quickly saturates the radio medium, i.e. one 100-byte frame every 12.5ms
- 2/ only 1-hop communication



Xbee GW

GW also in AP0 mode

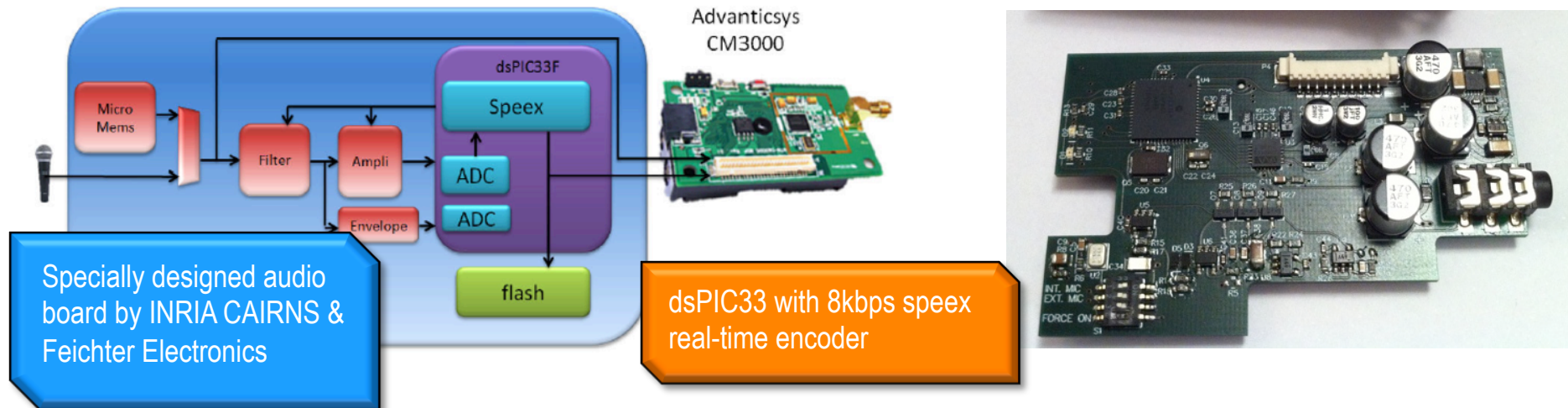
Use python script to read serial port /dev/ttyUSB0

```
> python SerialToStdout | play --buffer 50 -t raw -r 8000 -u -1 -
```



2nd solution: dev. of audio board

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

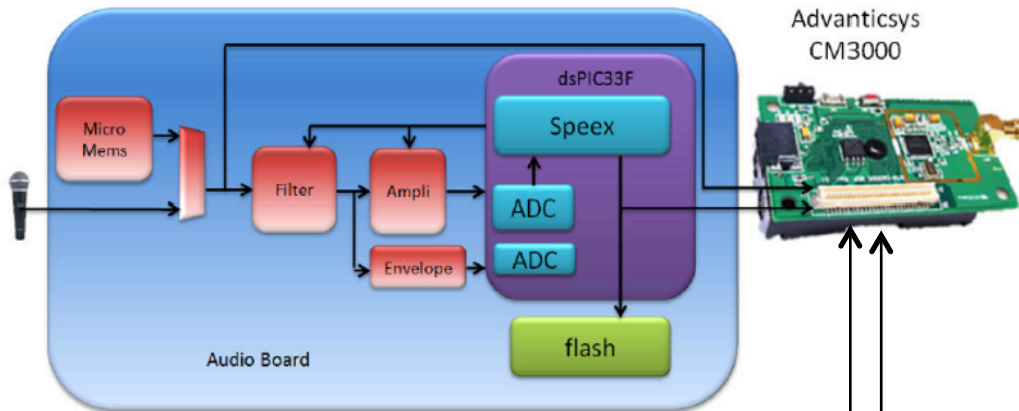


- Allows for multi-hop, encoded audio streaming scenarios

Audio board principle

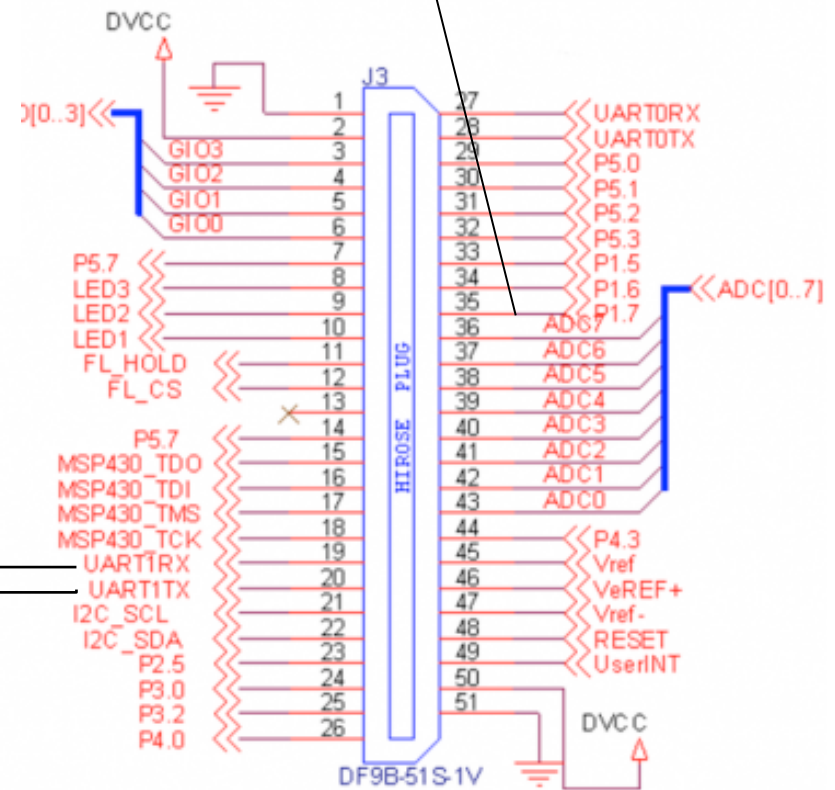
- The audio board captures 160 bytes (20ms) of raw audio and uses speex codec at 8kbps to produce 20 bytes to encoded audio data
- It sends the encoded audio data through an UART line to the host micro-controller
- The host micro-controller receives the encoded data and sends them wirelessly to the next hop
- The last hop is a base station that will forward the encoded audio into a speex audio decoder
- Output of the speex audio decoder is in raw format that can be feed into a player (`play`)

Audio board on TelosB

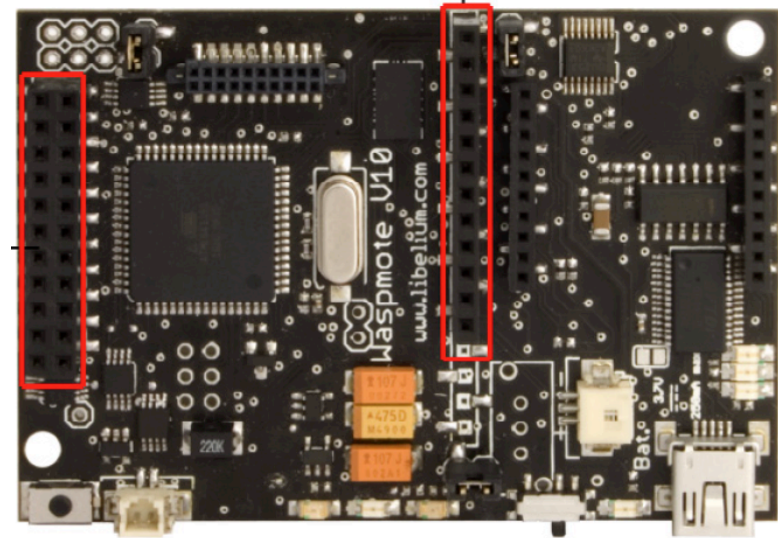
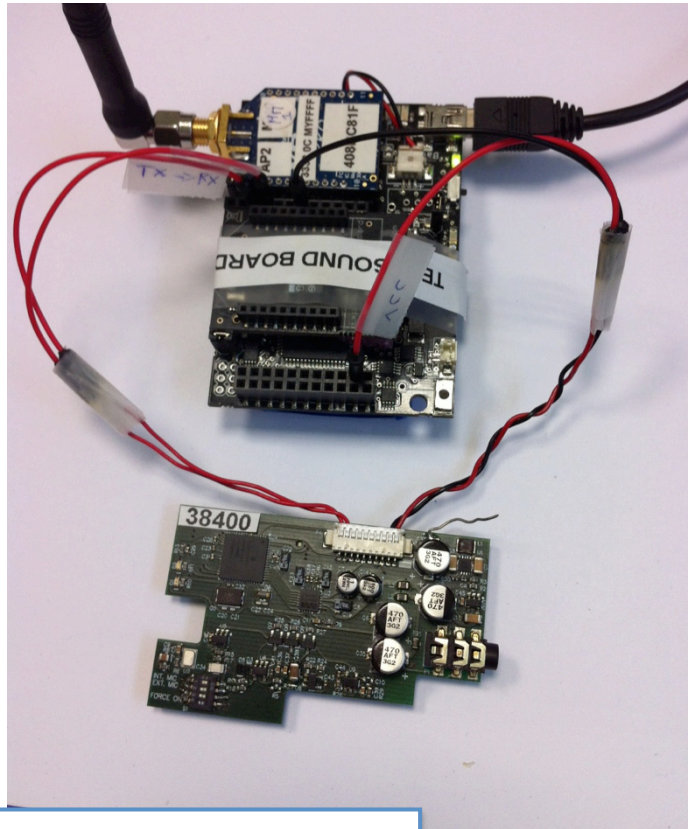


P1.7 can be used to power on/off the audio board

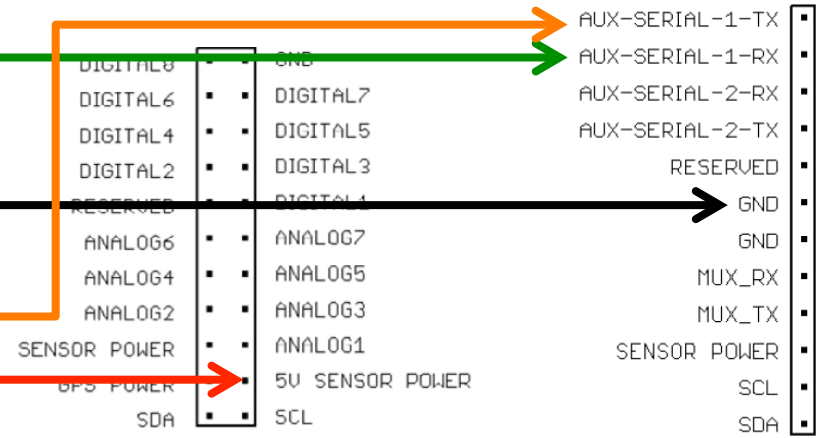
Initial development on Advanticsys TelosB mote. Encoded audio data transfers are realized with UART (serial) communication so easy to connect to other sensor boards.



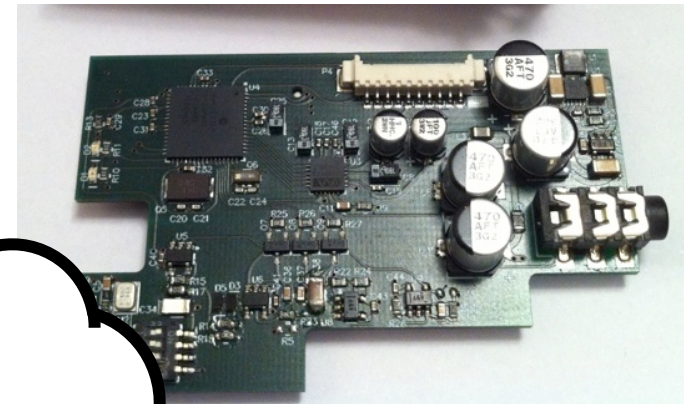
Audio board on WaspMote



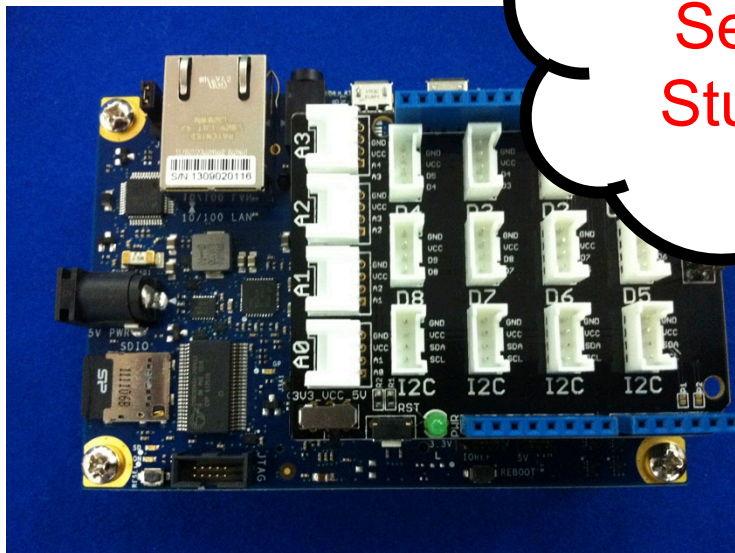
TX on AUX-SERIAL-1-RX
 (RX on AUX-SERIAL-1-TX)
 VCC on 5V
 GND on GND



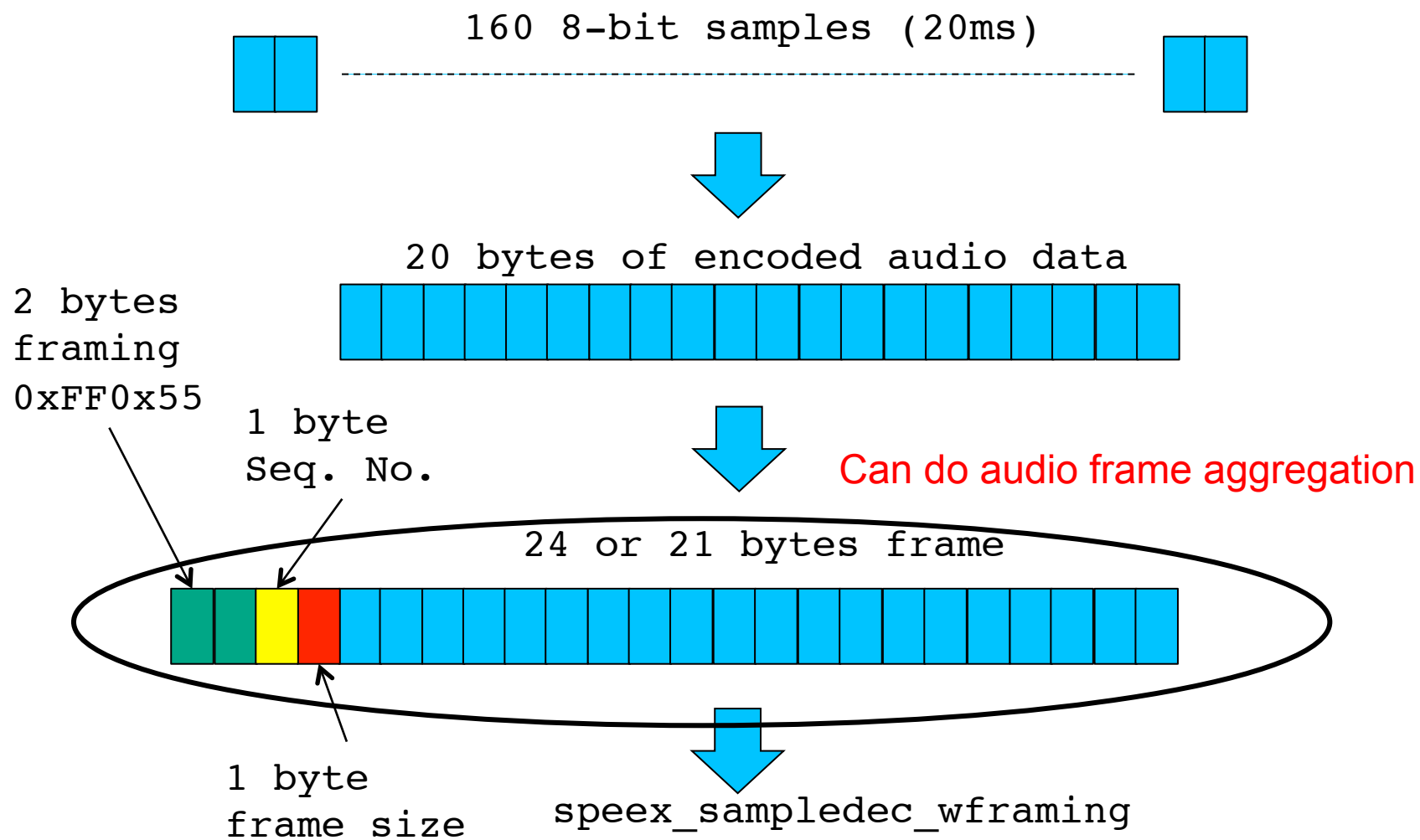
Audio board on Intel's Galileo



Senzation's
Sensational
Student team



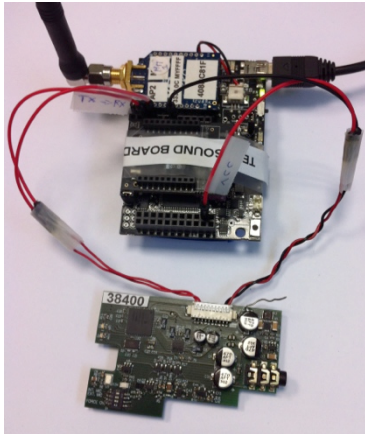
speex at 8kbps



Summary of audio constraints

Codec	Minimum sending rate
Raw 4KHz 8KHz	100 bytes every 25ms 100 bytes every 12.5ms
Speex 8000bps A1 A2 A3 A4	24 bytes every 20ms 48 bytes every 40ms 72 bytes every 60ms 96 bytes every 80ms

IoT node sending performance

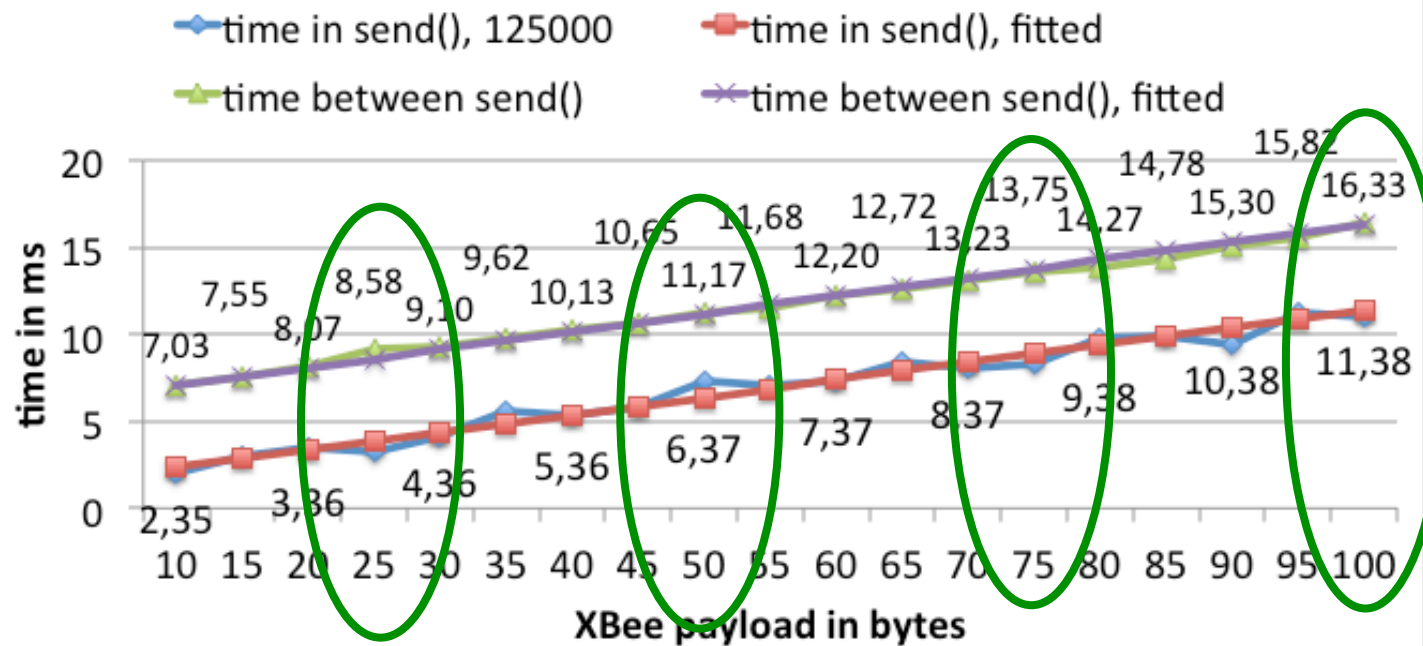


LIBELIUM WASPMOTE

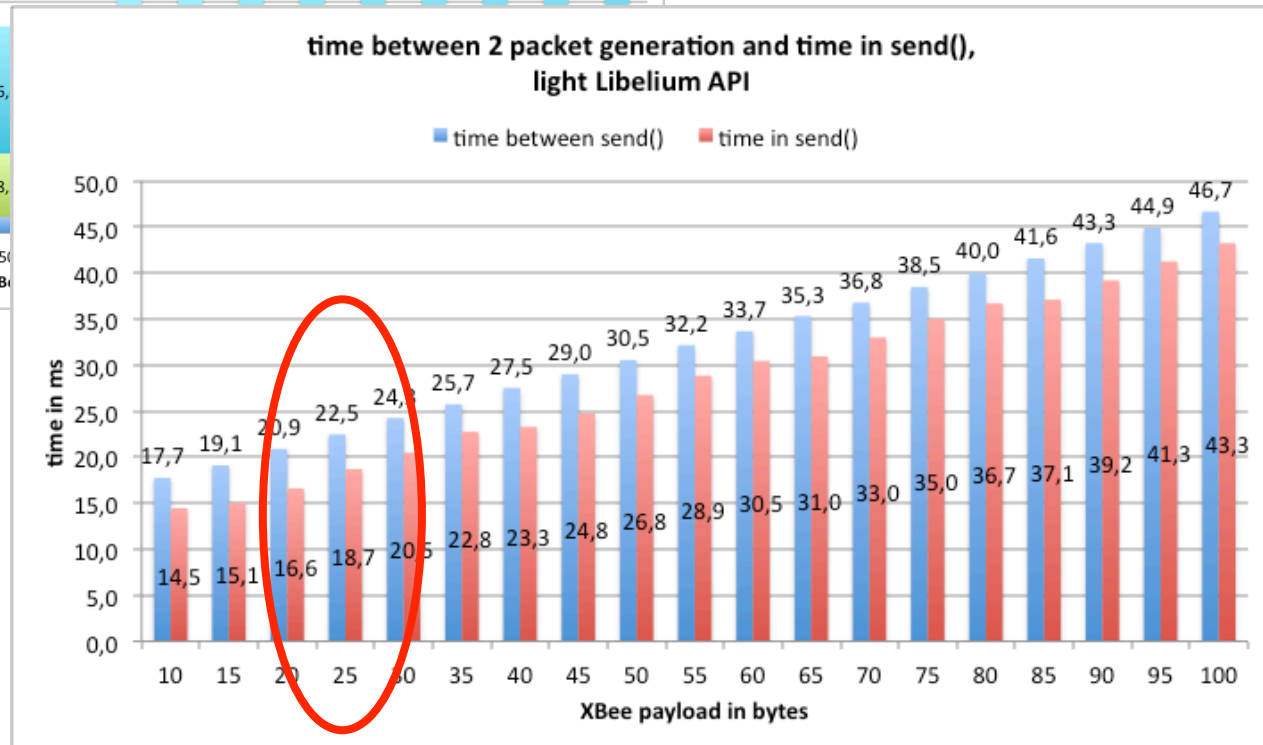
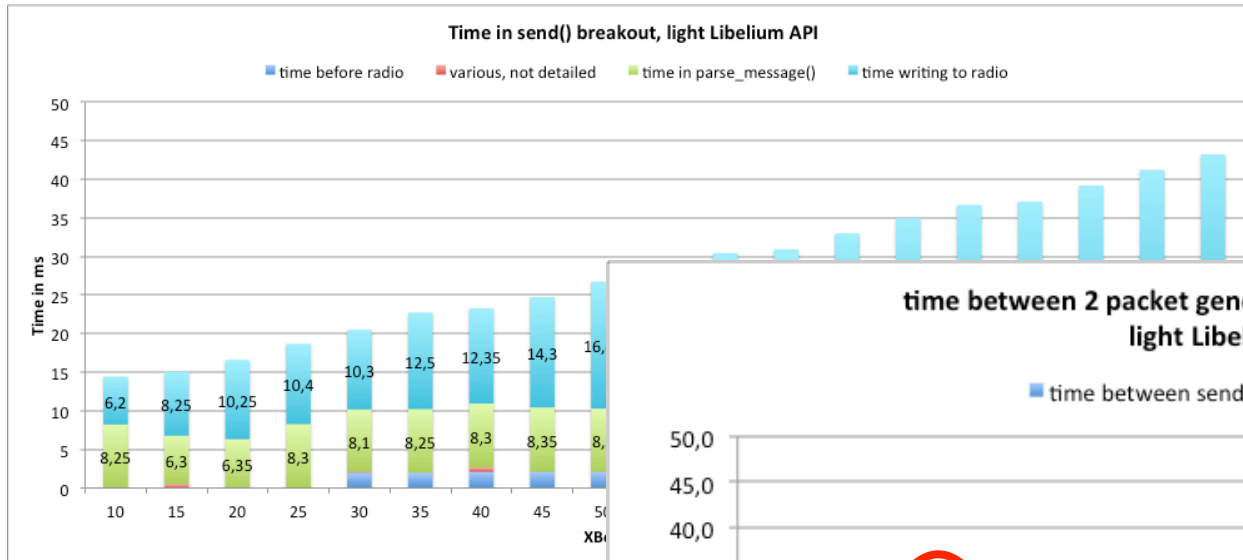
24 bytes every 20ms



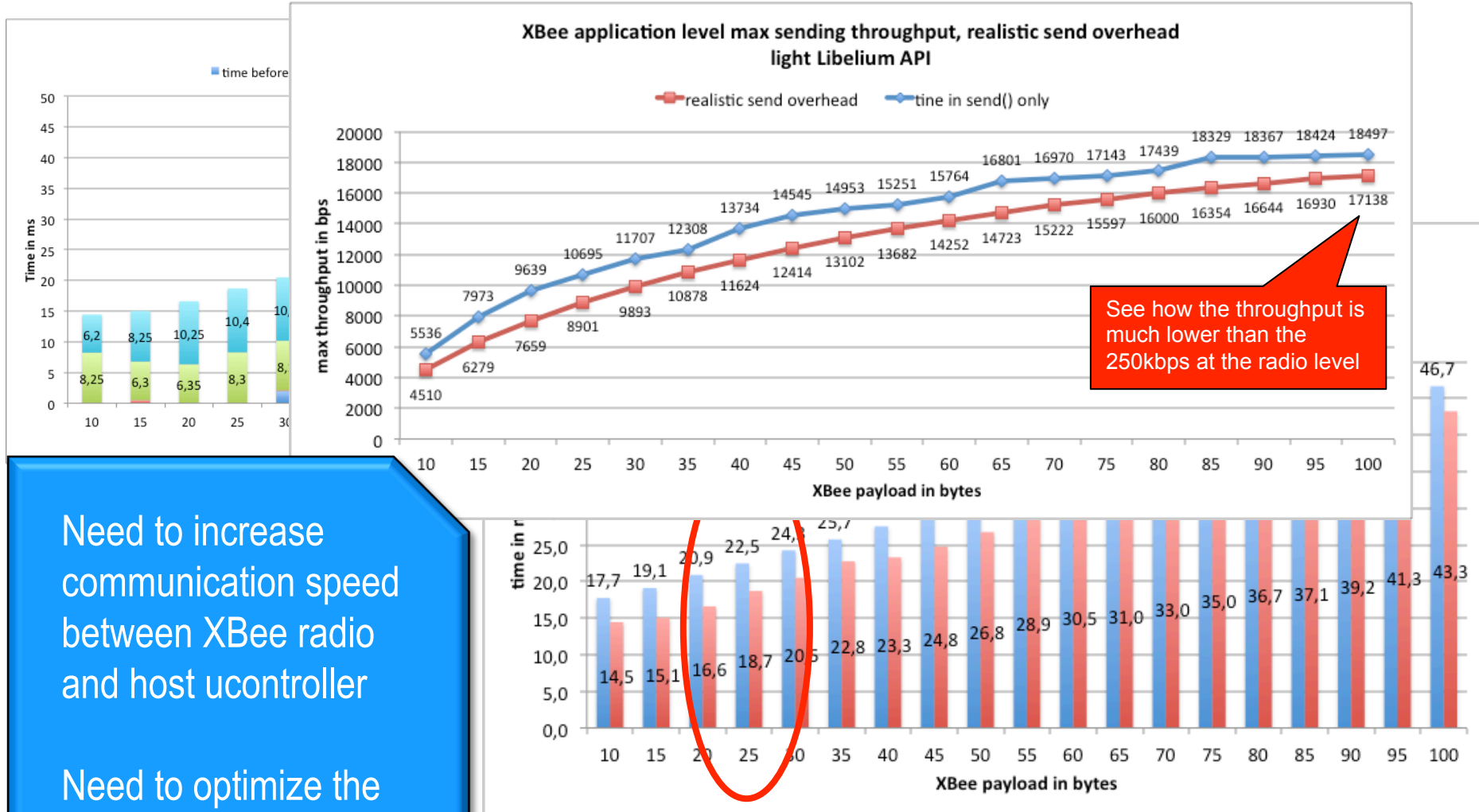
Time in send() and time between 2 packet generation
Libelium WaspMote



Off-the-shelves performance



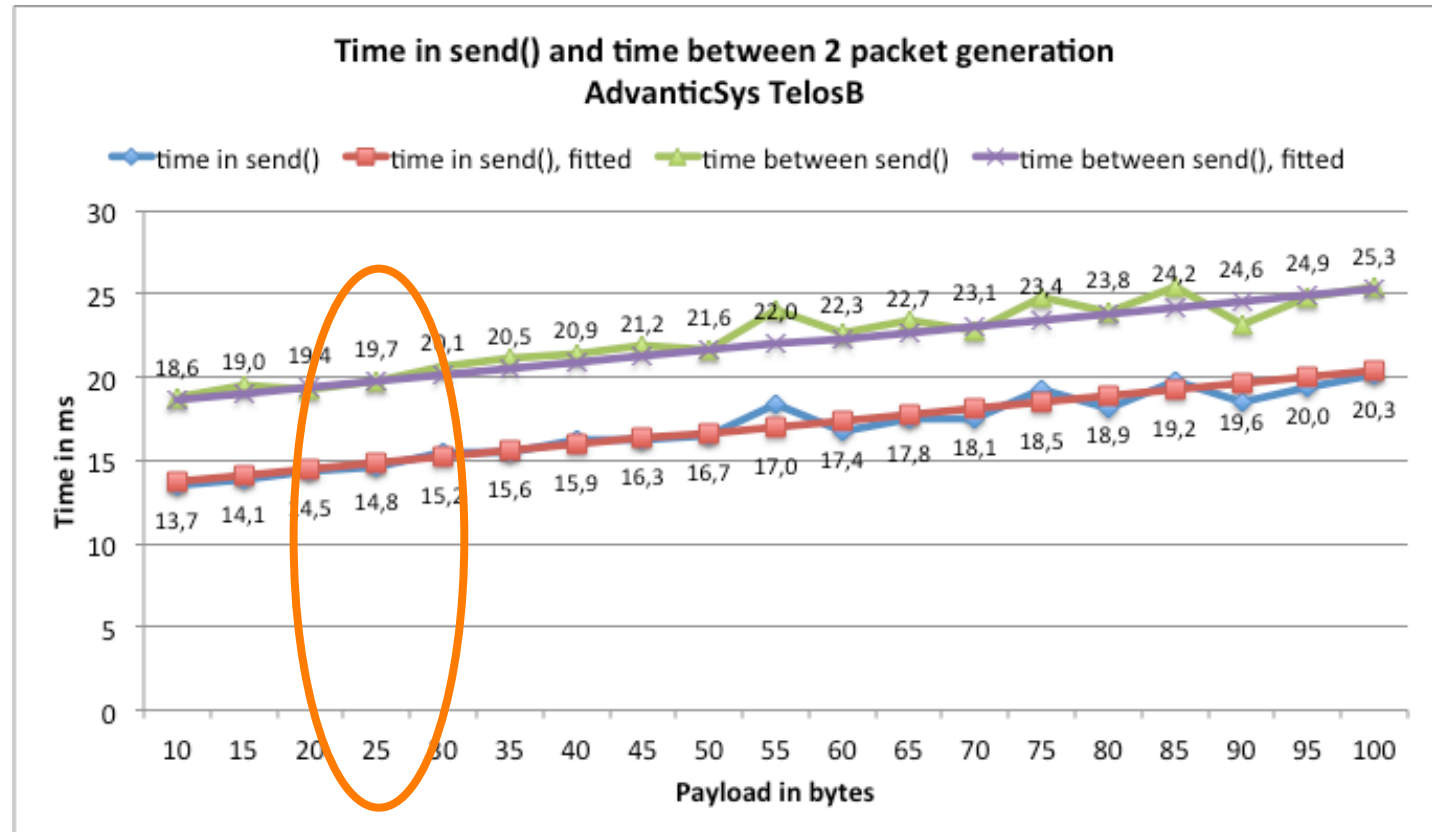
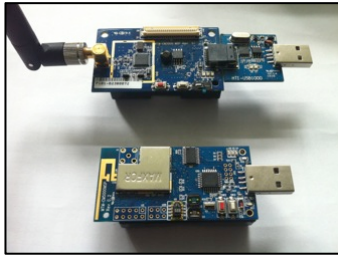
Off-the-shelves performance



Need to increase communication speed between XBee radio and host ucontroller

Need to optimize the API

IoT node sending performance



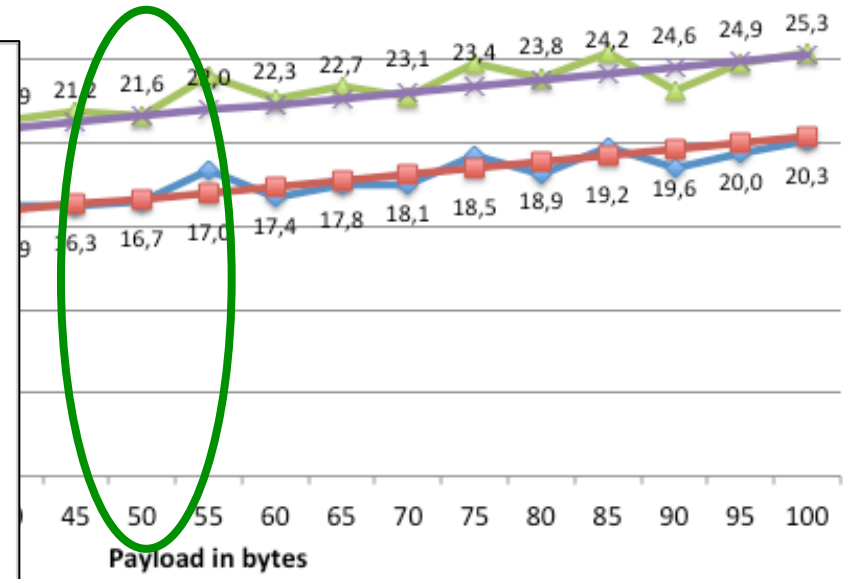
IoT node sending performance



Time in send() and time between 2 packet generation
AdvanticsSys TelosB

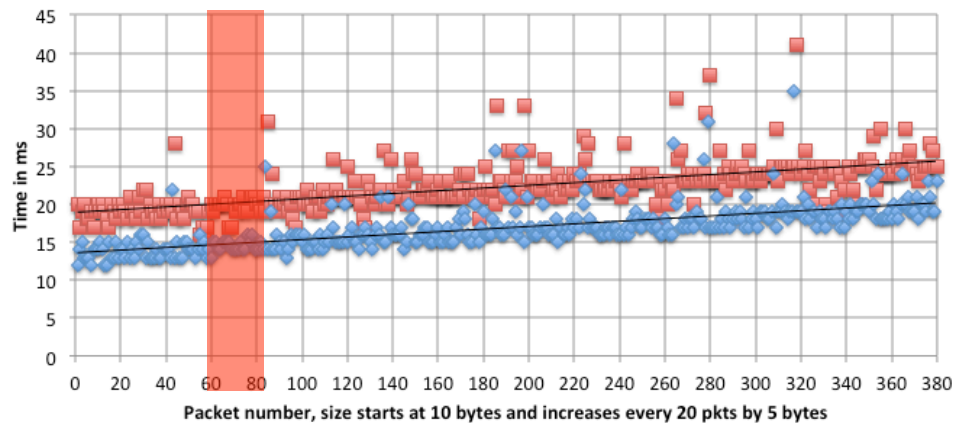
time in send() time in send(), fitted time between send() time between send(), fitted

30



Time in send() & time between 2 packet generation

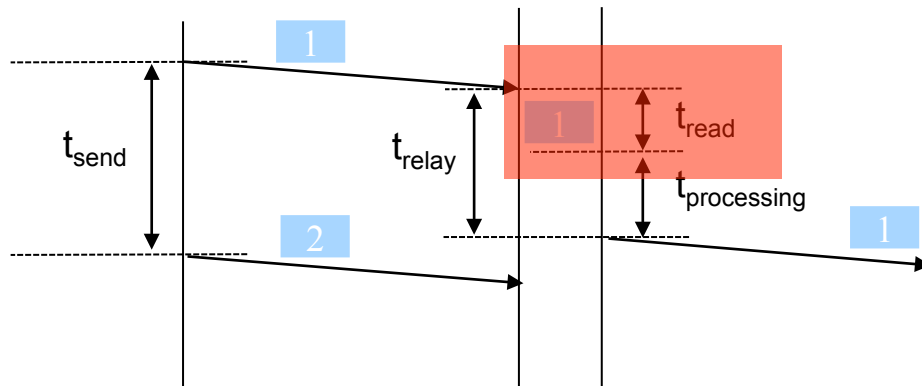
time between 2 packet generation Time in send()
Linéaire (time between 2 packet generation) Linéaire (Time in send())



Better with A2 agregation



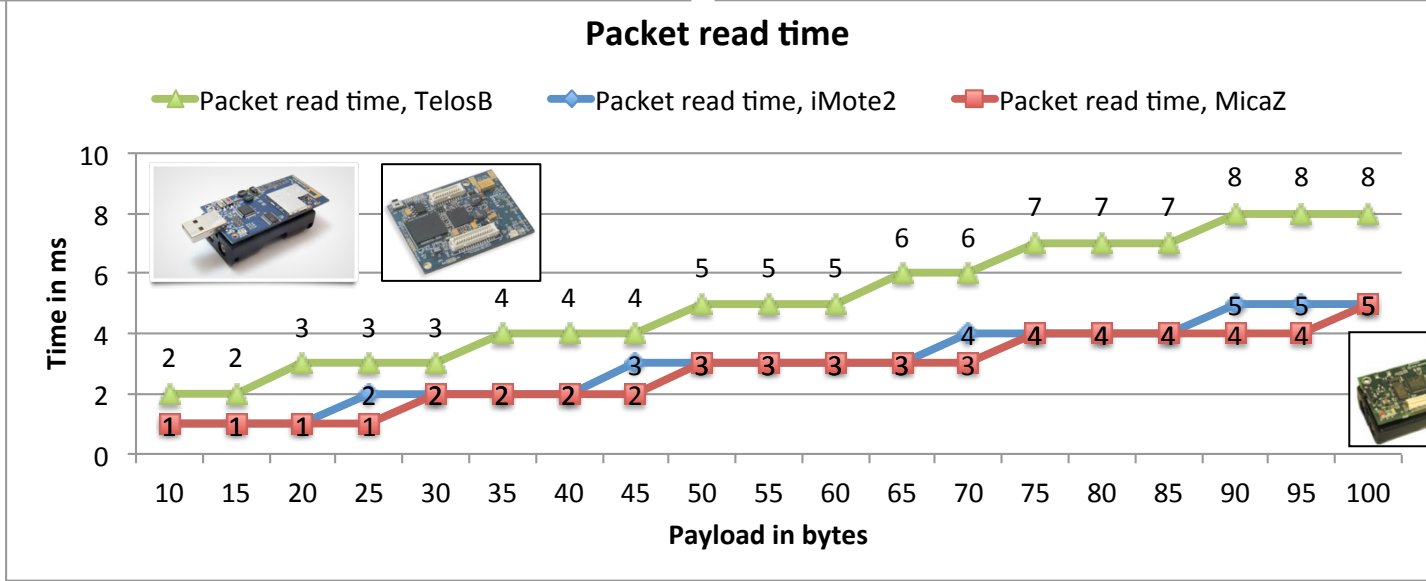
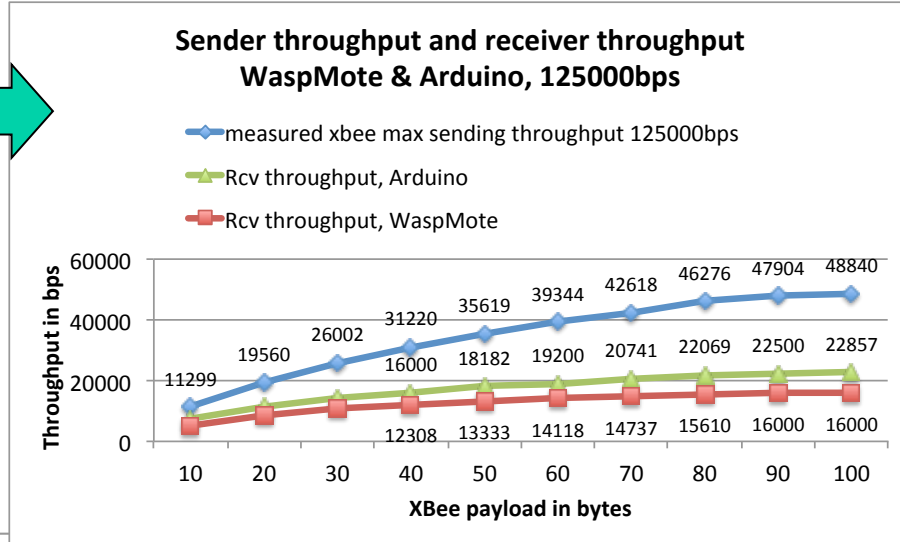
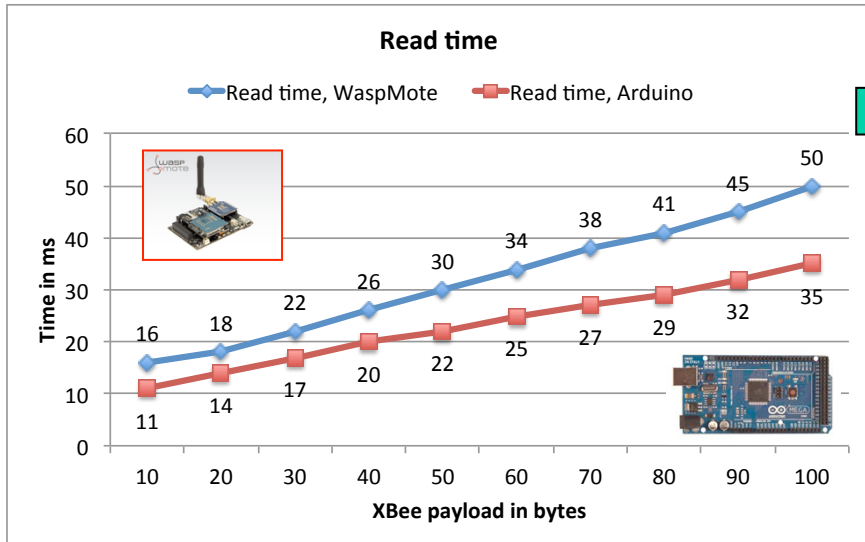
Receive performances



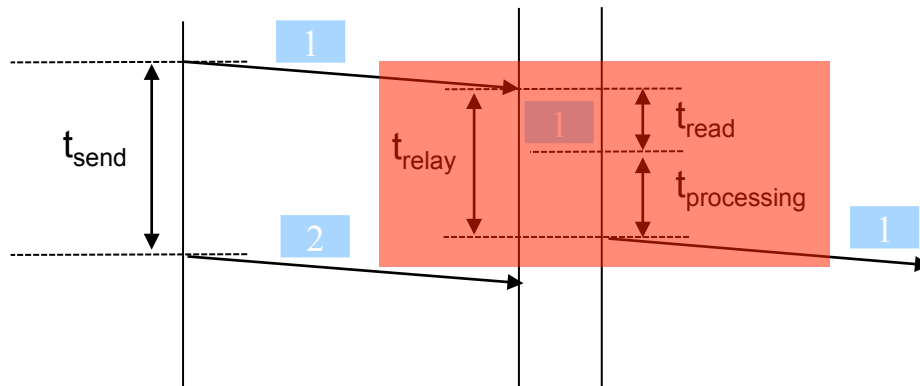
- At sender side, send as fast as possible
- At receiver side, determine t_{read}
- ... and also compute the maximum receive throughput per packet size



T_{READ} for various motes



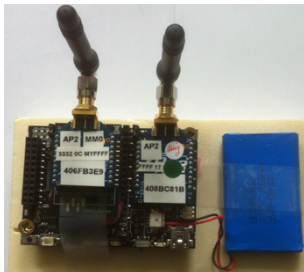
Relay performances



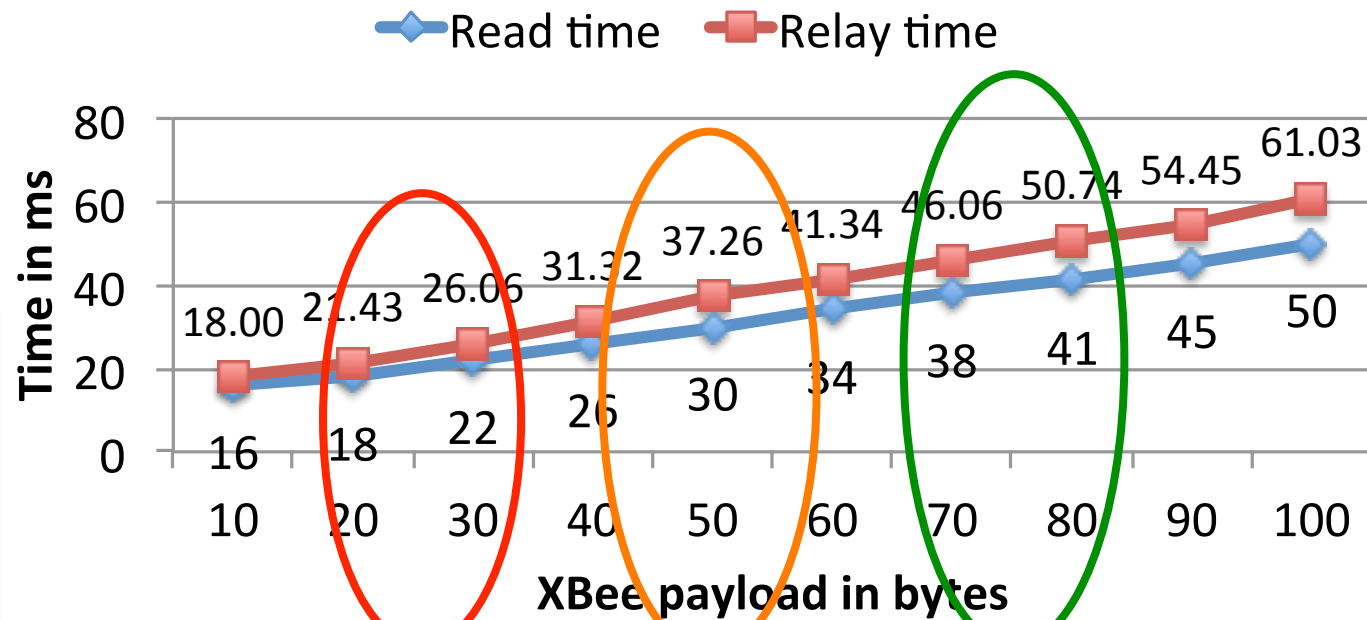
- Relaying are usually done at application-level (even OS level is considered app-level for the mote)
- Relaying means:
 - Read the packet in memory
 - Send the packet to next hop



Relay node performances



Pkt read time & Pkt relay time, WaspMote



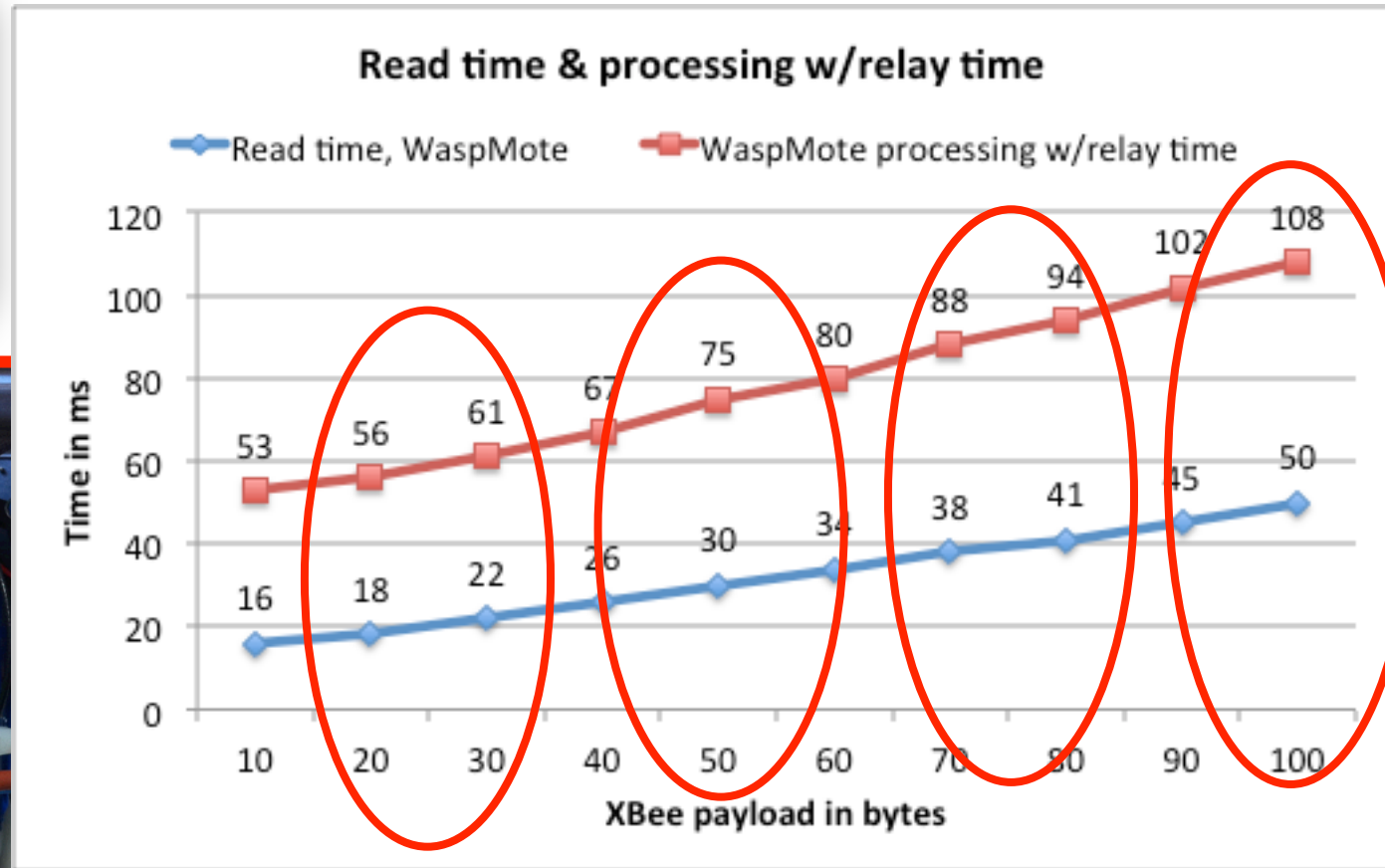
SPEEX codec at 8kbps requires to be able to relay a 25-byte packet every 20ms

Needs A3 agregation

Santander's limitations



SmartSantander's IoT node uses 38400 baud rate for communication between XBee radio and host ucontroller

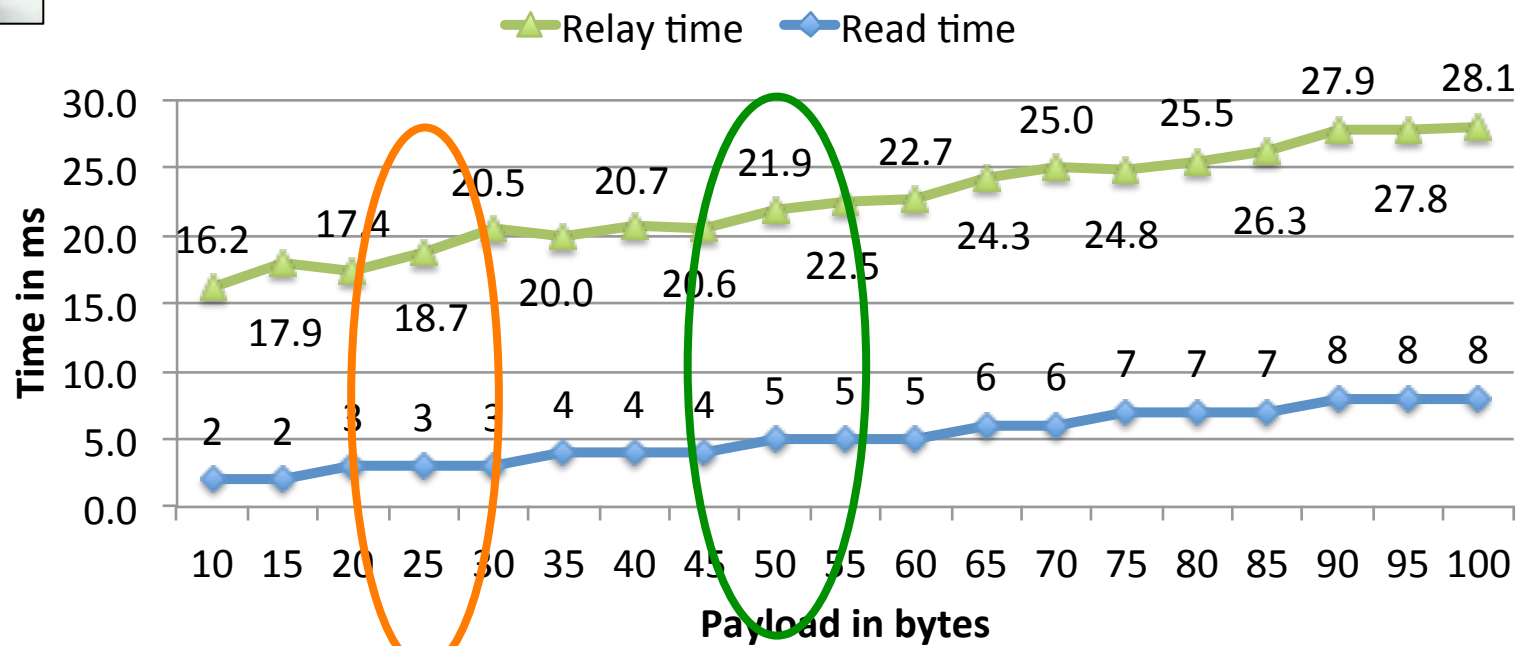


Needs to discard audio frame at the source to increase the time window

Relay node performances



Pkt read time & Pkt relay time, TelosB



Needs A2 agregation



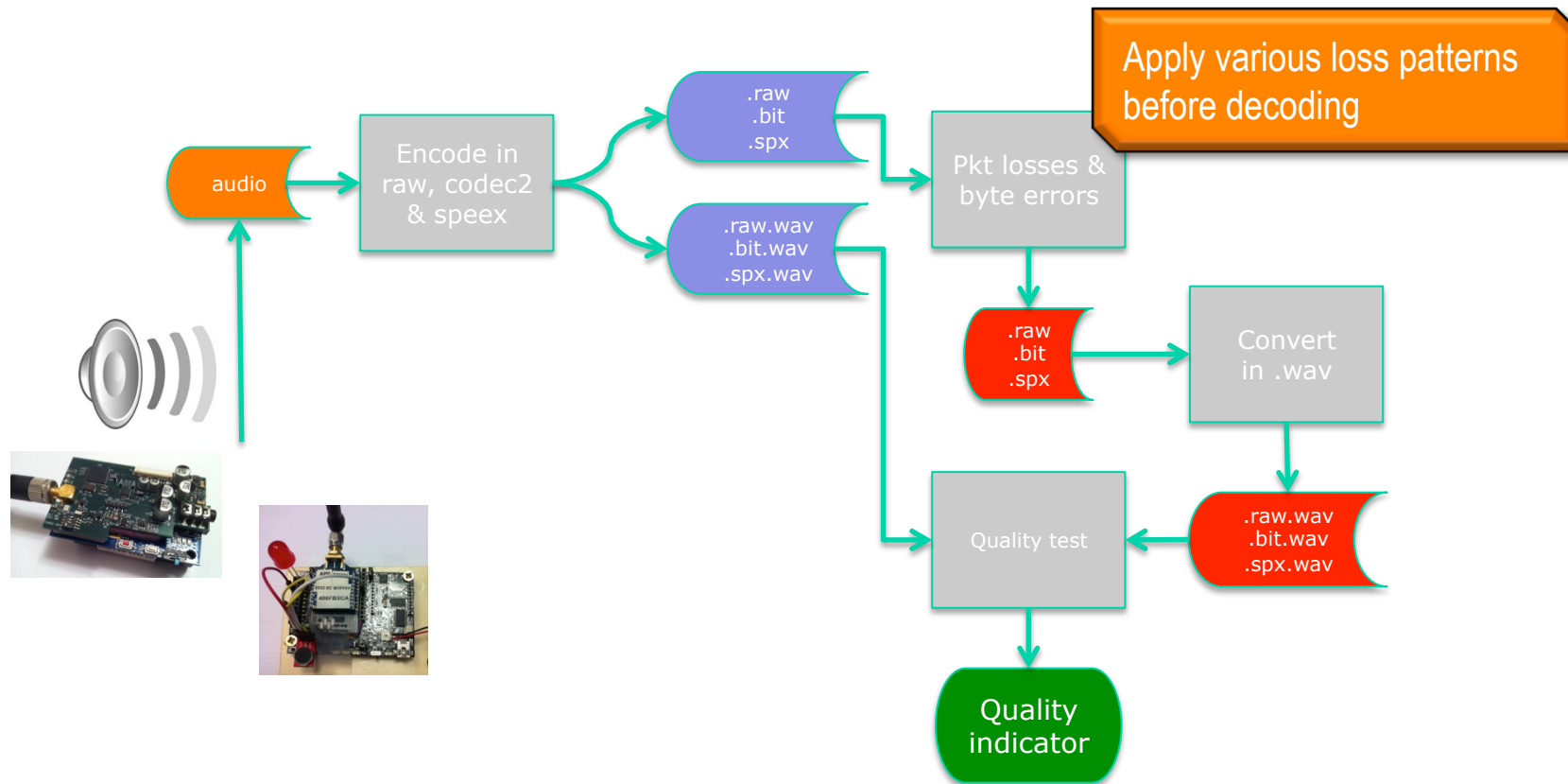
Summary on performances



- Looking at app-level performances, taking into account OS&API overheads (read, write)
- In multi-hop communication, relay time can dramatically reduce the E2E performances
- Dedicated audio board leverage some performance limits
- Audio frame aggregation can be used to adapt to IoT node limits



Sensitivity of codecs



Audio quality: PESQ & MOS (1)

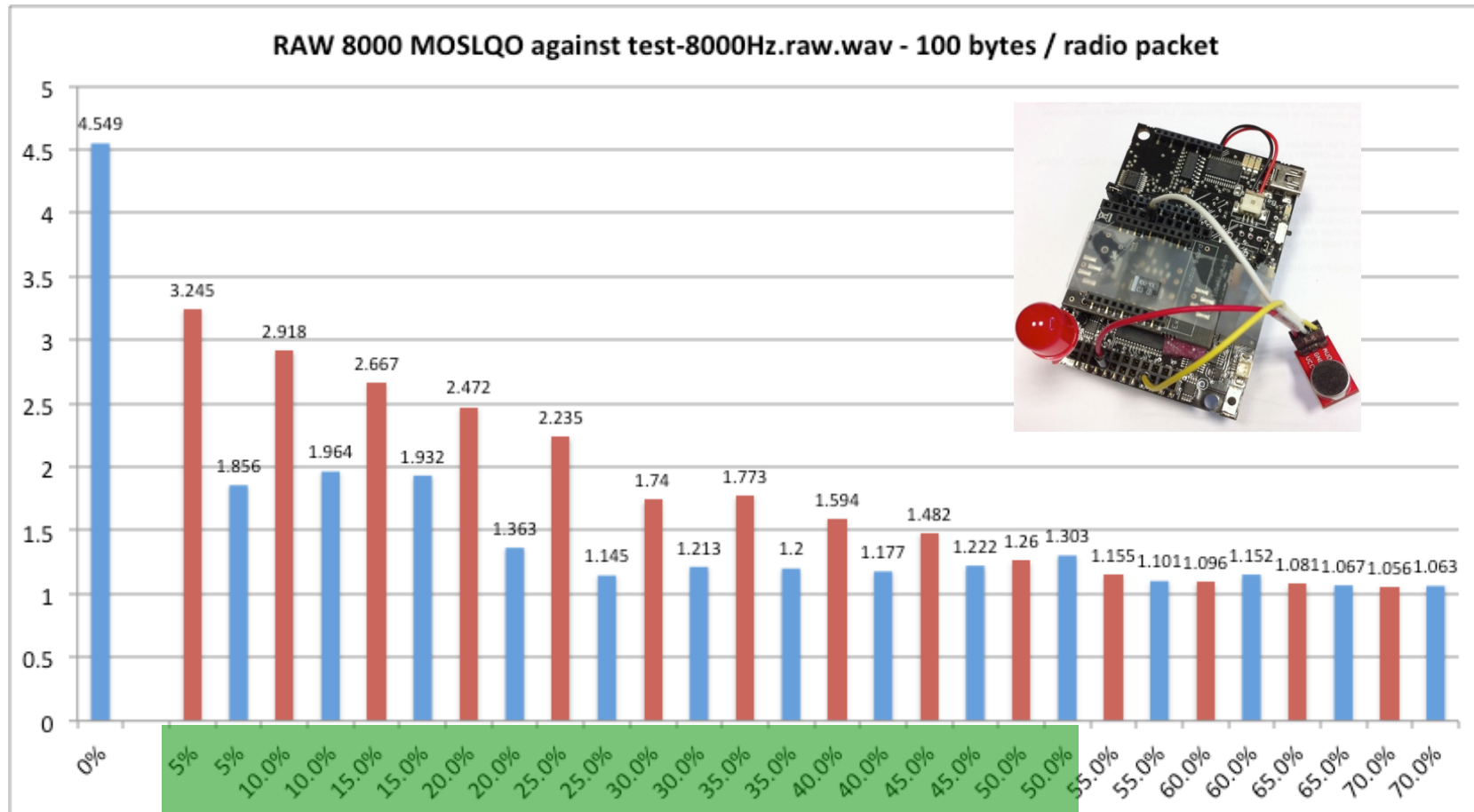
- ITU-T P.862 Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs.
- We can use ITU-T PESQ tool to determine the MOS value for loss-free encoded audio (codec2, speex, ...). MOS-LQO values greater than 2.6 are considered good.

Audio quality: PESQ & MOS (2)

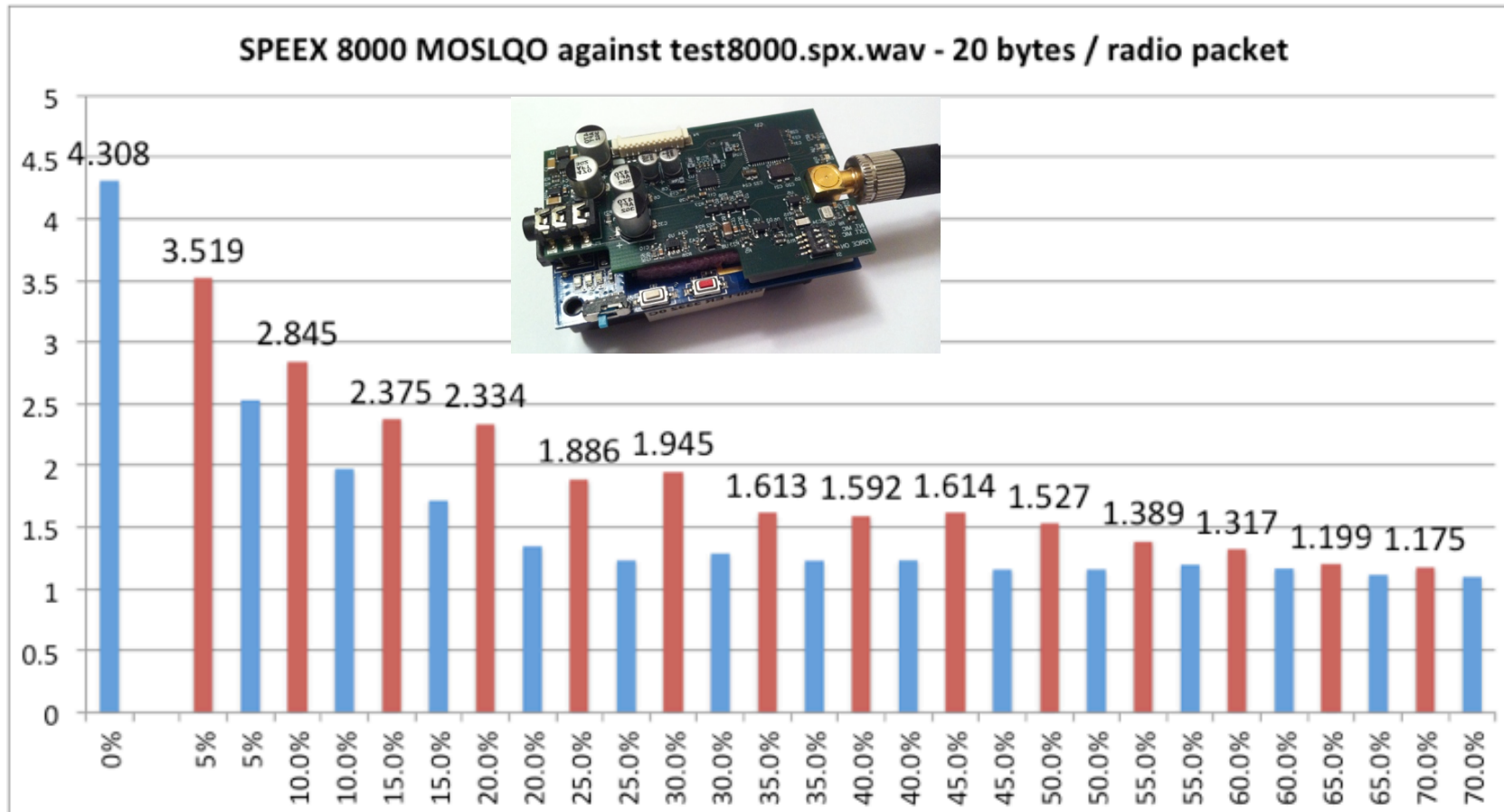
- 5=Excellent, 4=Good, 3=Fair, 2=Poor, 1=Bad

REFERENCE	DEGRADED	PESQMOS	MOSLQO	SAMPLE_FREQ
test.wav	test.wav	4.500	4.549	8000
test.wav	test4000Hz.raw.wav	0.769	1.115	4000
test.wav	test8000Hz.raw.wav	4.5	4.549	8000
test.wav	test2150.spx.wav	2.757	2.472	8000
test.wav	test5950.spx.wav	3.428	3.454	8000
test.wav	test8000.spx.wav	3.652	3.757	8000
test.wav	test11000.spx.wav	3.941	4.093	8000
test.wav	test13000.spx.wav	3.941	4.093	8000
test.wav	test15000.spx.wav	4.085	4.235	8000

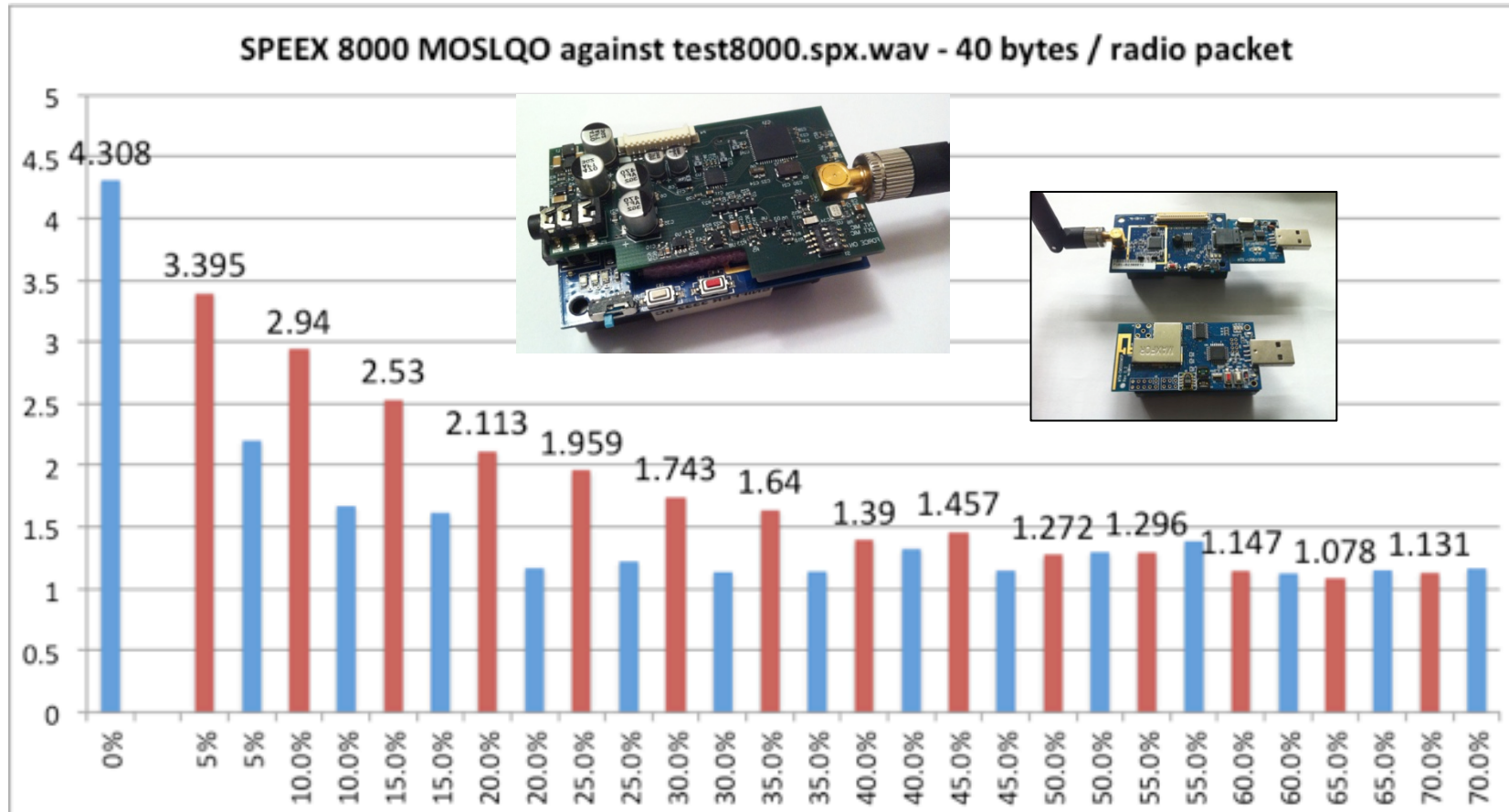
Test8000.raw



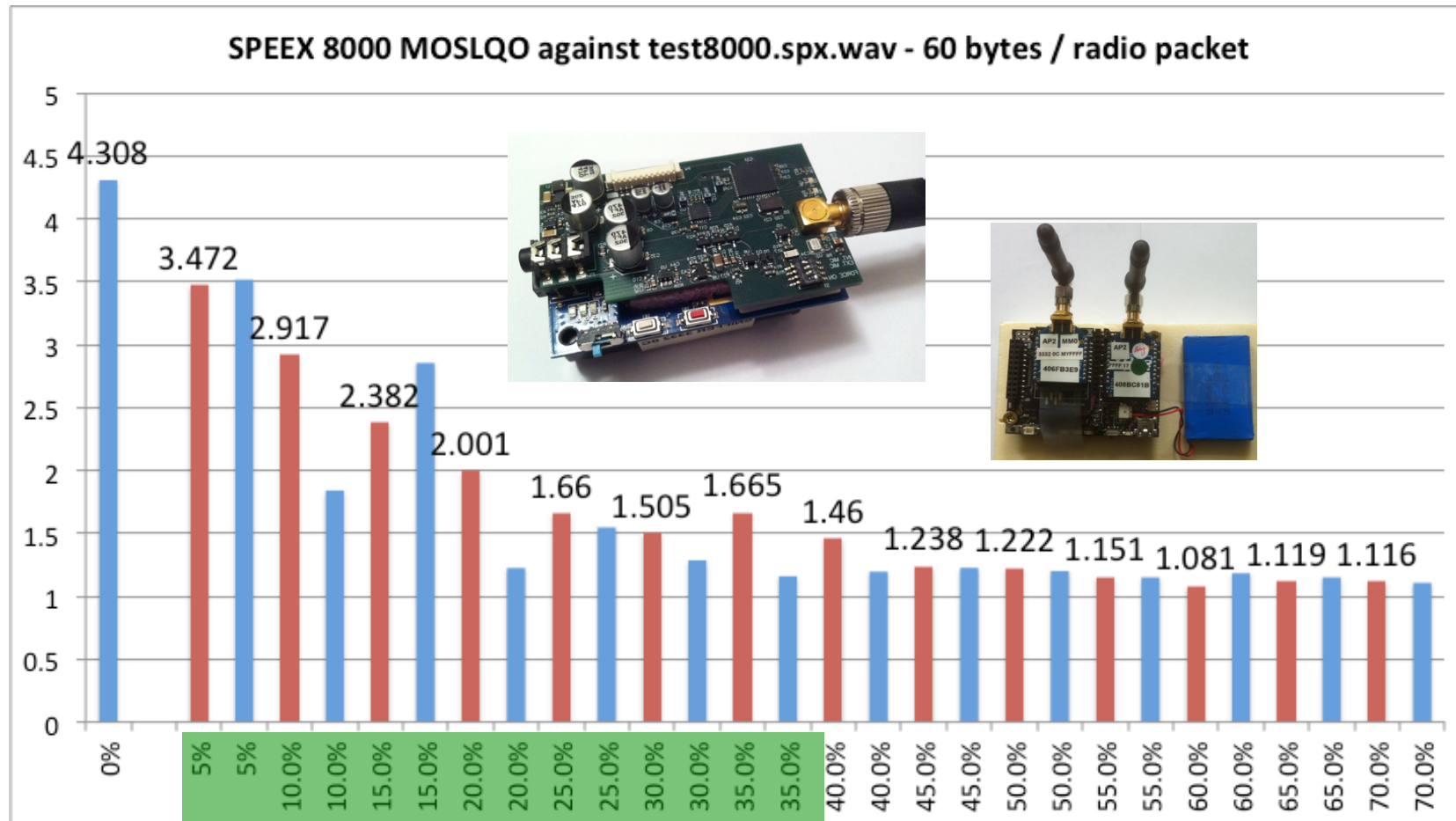
Test8000.spx, 20B/pkt (A1)



Test800.spx, 40B/pkt (A2)

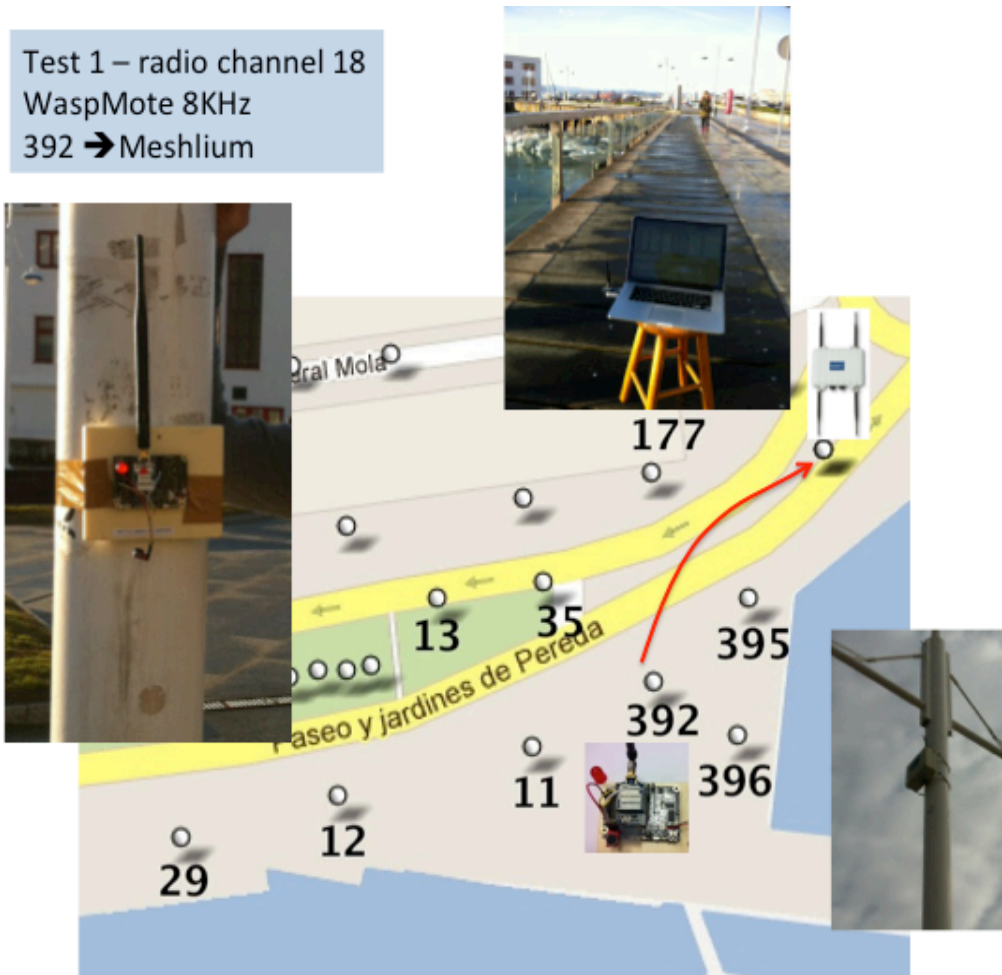


Test800.spx, 60B/pkt (A3)



1-hop audio in LoS

Test 1 – radio channel 18
WaspMote 8KHz
392 → Meshlium



Frame analysis

- Use `wireshark` as frame analysis tool
- AdvanticSys TelosB mote as promiscuous sniffer mote, connected to `wireshark` to display captured frames
- Frame reception time can be visualized for statistic collection
 - Transmission latencies
 - Frame jitter



the sounds of smart environments



wireshark frame capture



audio_capture [Wireshark 1.6.7]

Filter: Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Sequence Number	Extra info	Data
23	68719.476721			IEEE 802.15.4	5		77 68576.10478	
24	150.135872	00:13:a2:00:40:92:20:70	0x0090	IEEE 802.15.4	22		78 - 68569.3408	Yes
25	68719.476721			IEEE 802.15.4	5		78 68569.34084	
26	*REF*	0x0090	0x0100	IEEE 802.15.4	35		144 *REF*	Yes
27	0.019584	0x0090	0x0100	IEEE 802.15.4	35		145 0.019584	Yes
28	0.047456	0x0090	0x0100	IEEE 802.15.4	35		146 0.027872	Yes
29	0.061824	0x0090	0x0100	IEEE 802.15.4	35		147 0.014368	Yes
30	0.083456	0x0090	0x0100	IEEE 802.15.4	35		148 0.021632	Yes
31	0.103584	0x0090	0x0100	IEEE 802.15.4	35		149 0.020128	Yes
32	0.128064	0x0090	0x0100	IEEE 802.15.4	35		150 0.024480	Yes
33	0.147104	0x0090	0x0100	IEEE 802.15.4	35		151 0.019040	Yes
34	0.167872	0x0090	0x0100	IEEE 802.15.4	35		152 0.020768	Yes
35	0.187072	0x0090	0x0100	IEEE 802.15.4	35		153 0.019200	Yes
36	0.210752	0x0090	0x0100	IEEE 802.15.4	35		154 0.023680	Yes
37	0.229952	0x0090	0x0100	IEEE 802.15.4	35		155 0.019200	Yes
38	0.249792	0x0090	0x0100	IEEE 802.15.4	35		156 0.019840	Yes
39	0.274880	0x0090	0x0100	IEEE 802.15.4	35		157 0.025088	Yes
40	0.290816	0x0090	0x0100	IEEE 802.15.4	35		158 0.015936	Yes
41	0.312224	0x0090	0x0100	IEEE 802.15.4	35		159 0.021408	Yes
42	0.333952	0x0090	0x0100	IEEE 802.15.4	35		160 0.021728	Yes

▼ Frame 26: 35 bytes on wire (280 bits), 35 bytes captured (280 bits)
 Arrival Time: Dec 31, 1969 16:02:30.684992000 PST
 Epoch Time: 150.684992000 seconds
 [Time delta from previous captured frame: -68568.791728000 seconds]
 [Time delta from previous displayed frame: -68568.791728000 seconds]
 [Time since reference or first frame: 0.000000000 seconds]
 [This is a Time Reference frame]
 Frame Number: 26
 Frame Length: 35 bytes (280 bits)
 Capture Length: 35 bytes (280 bits)
 [Frame is marked: False]
 [Frame is ignored: False]
 [Protocols in frame: wlan:data]

▼ IEEE 802.15.4 Data, Dst: 0x0100, Src: 0x0090, Bad FCS

- ▶ Frame Control Field: Data (0x0841)
 - Sequence Number: 144
 - Destination PAN: 0x3332
 - Destination: 0x0100
 - Source: 0x0090
 - FCS: 0xffff (Incorrect, expected FCS=0xa563)
 - [Expert Info (Warn/Checksum): Bad FCS]
- ▶ Data (24 bytes)

```

0000 41 88 90 32 33 00 01 90 00 ff 55 01 14 ae 24 24  A..23... ..U...$$
0010 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24  $$$$$$$$ $$$$$$$$
0020 24 ff ff  $..

```

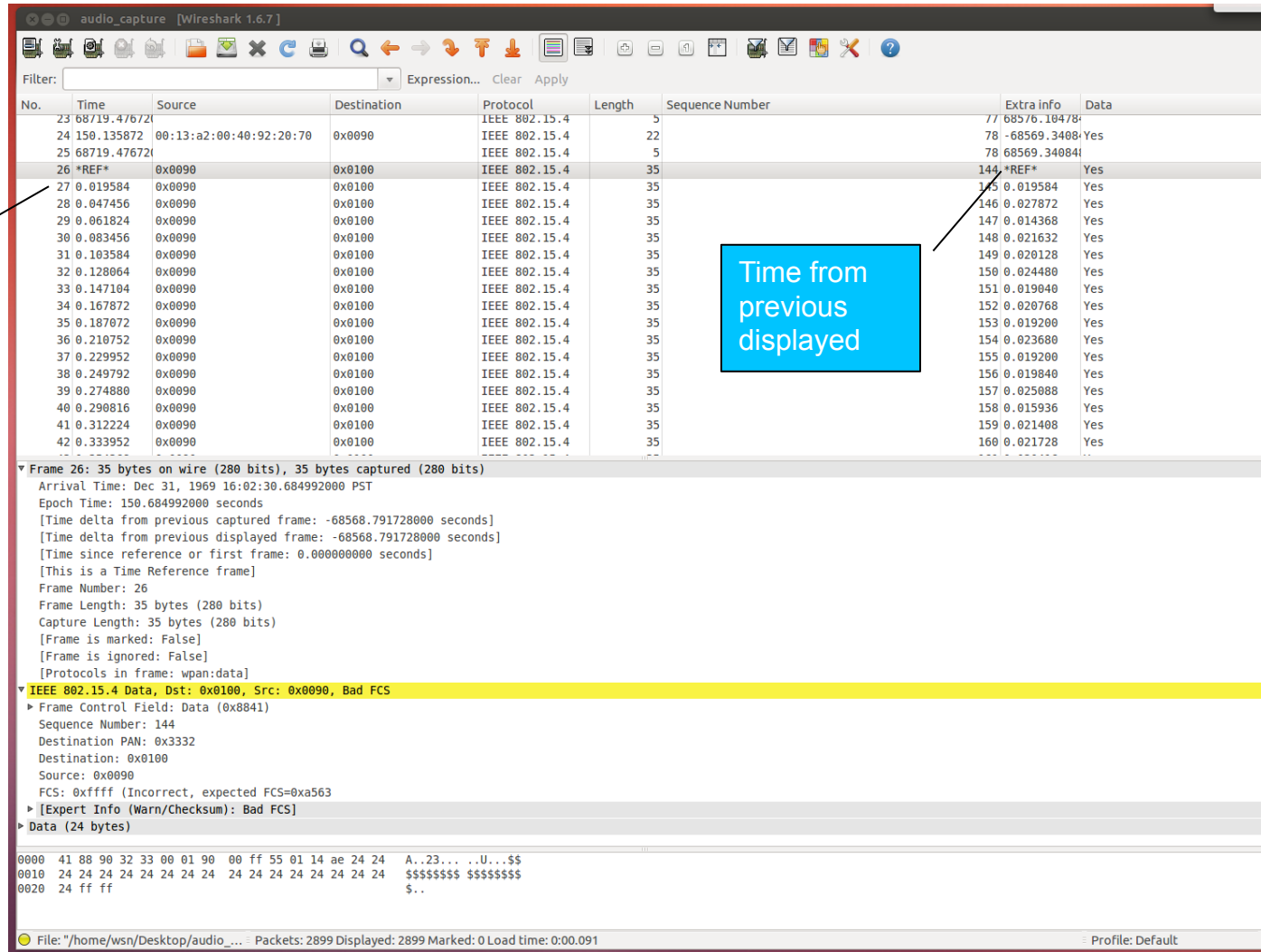
File: "/home/wsn/Desktop/audio_... Packets: 2899 Displayed: 2899 Marked: 0 Load time: 0:00.091 Profile: Default

Example: latency 1-hop



Time from reference time

Time from previous displayed



No.	Time	Source	Destination	Protocol	Length	Sequence Number	Extra info	Data
23	68719.476721			IEEE 802.15.4	5		77 68576.10478	
24	150.135872	00:13:a2:00:40:92:20:70	0x0090	IEEE 802.15.4	22		78 -68569.3408	Yes
25	68719.476721			IEEE 802.15.4	5		78 68569.34084	
26	*REF*	0x0090	0x0100	IEEE 802.15.4	35		144 *REF*	Yes
27	0.019584	0x0090	0x0100	IEEE 802.15.4	35		145 0.019584	Yes
28	0.047456	0x0090	0x0100	IEEE 802.15.4	35		146 0.027872	Yes
29	0.061824	0x0090	0x0100	IEEE 802.15.4	35		147 0.014368	Yes
30	0.083456	0x0090	0x0100	IEEE 802.15.4	35		148 0.021632	Yes
31	0.103584	0x0090	0x0100	IEEE 802.15.4	35		149 0.020128	Yes
32	0.128064	0x0090	0x0100	IEEE 802.15.4	35		150 0.024480	Yes
33	0.147104	0x0090	0x0100	IEEE 802.15.4	35		151 0.019040	Yes
34	0.167872	0x0090	0x0100	IEEE 802.15.4	35		152 0.020768	Yes
35	0.187072	0x0090	0x0100	IEEE 802.15.4	35		153 0.019200	Yes
36	0.210752	0x0090	0x0100	IEEE 802.15.4	35		154 0.023680	Yes
37	0.229952	0x0090	0x0100	IEEE 802.15.4	35		155 0.019200	Yes
38	0.249792	0x0090	0x0100	IEEE 802.15.4	35		156 0.019840	Yes
39	0.274880	0x0090	0x0100	IEEE 802.15.4	35		157 0.025088	Yes
40	0.290816	0x0090	0x0100	IEEE 802.15.4	35		158 0.015936	Yes
41	0.312224	0x0090	0x0100	IEEE 802.15.4	35		159 0.021408	Yes
42	0.333952	0x0090	0x0100	IEEE 802.15.4	35		160 0.021728	Yes

```

Frame 26: 35 bytes on wire (280 bits), 35 bytes captured (280 bits) on interface 0
Arrival Time: Dec 31, 1969 16:02:30.684992000 PST
Epoch Time: 150.684992000 seconds
[Time delta from previous captured frame: -68568.791728000 seconds]
[Time delta from previous displayed frame: -68568.791728000 seconds]
[Time since reference or first frame: 0.000000000 seconds]
[This is a Time Reference frame]
Frame Number: 26
Frame Length: 35 bytes (280 bits)
Capture Length: 35 bytes (280 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: wlan:data]

IEEE 802.15.4 Data, Dst: 0x0100, Src: 0x0090, Bad FCS
  Frame Control Field: Data (0x8841)
  Sequence Number: 144
  Destination PAN: 0x3332
  Destination: 0x0100
  Source: 0x0090
  FCS: 0xffff (Incorrect, expected FCS=0xa563)
  [Expert Info (Warn/Checksum): Bad FCS]
  Data (24 bytes)
    0000 41 88 90 32 33 00 01 90 00 ff 55 01 14 ae 24 24  A..23...U...$
    0010 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24  $$$$$$$$
    0020 24 ff ff                                     $..
  
```

File: "/home/wsn/Desktop/audio_... : Packets: 2899 Displayed: 2899 Marked: 0 Load time: 0:00.091 Profile: Default



Example: latency 2-hop



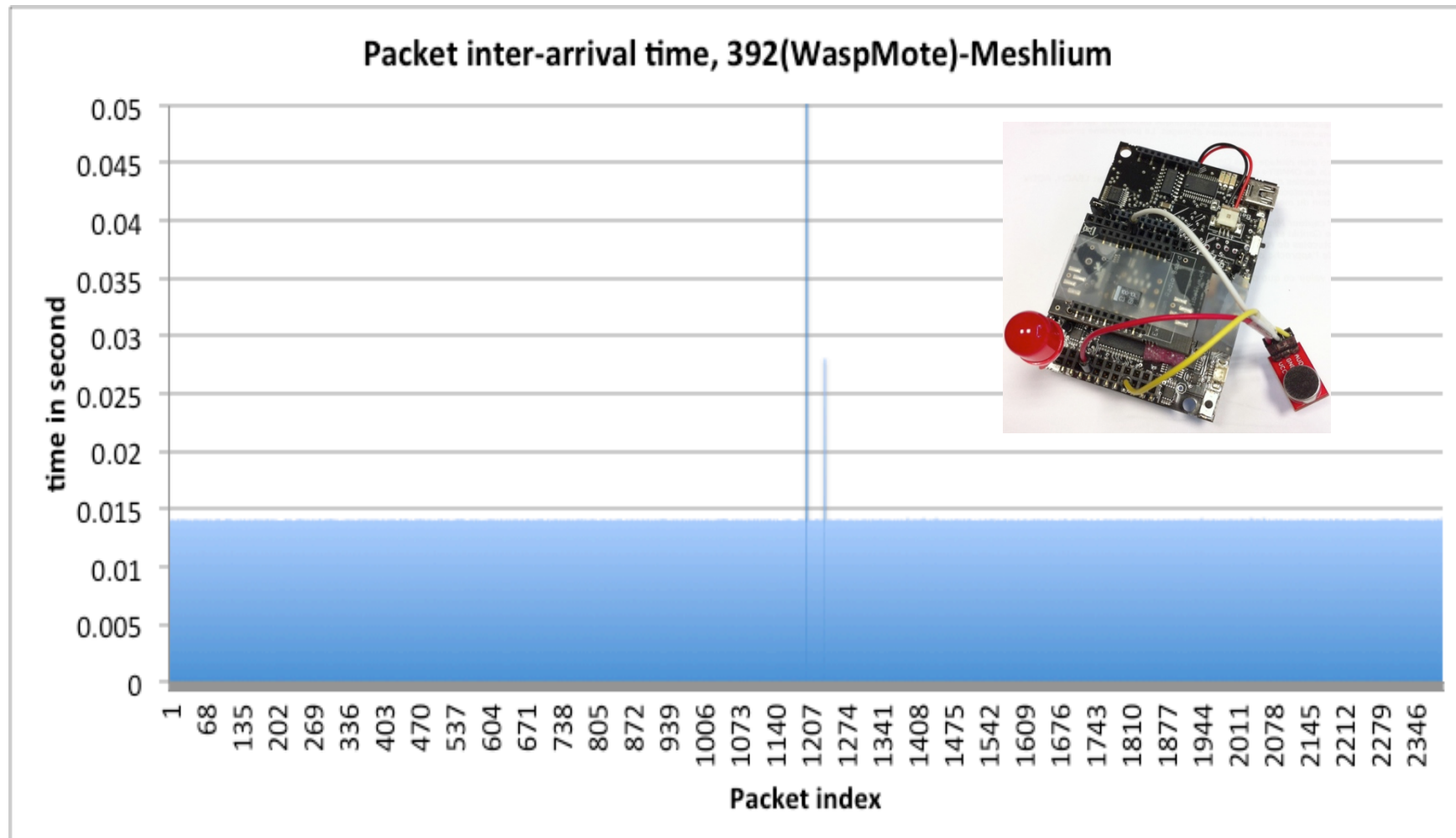
The image displays two screenshots of the Wireshark 1.6.7 network protocol analyzer. Both screenshots show a packet capture with the filter 'wpan.frame_type == 0x0001' applied.

Left Screenshot: Shows a list of captured packets. Packet 2238 is highlighted in red, indicating a reference frame. The packet details pane shows the IEEE 802.15.4 Data section with a destination of 0xc823 and source of 0x0090. The Frame Control Field is expanded, showing a Frame Type of 0x0001 and Security Enabled: False.

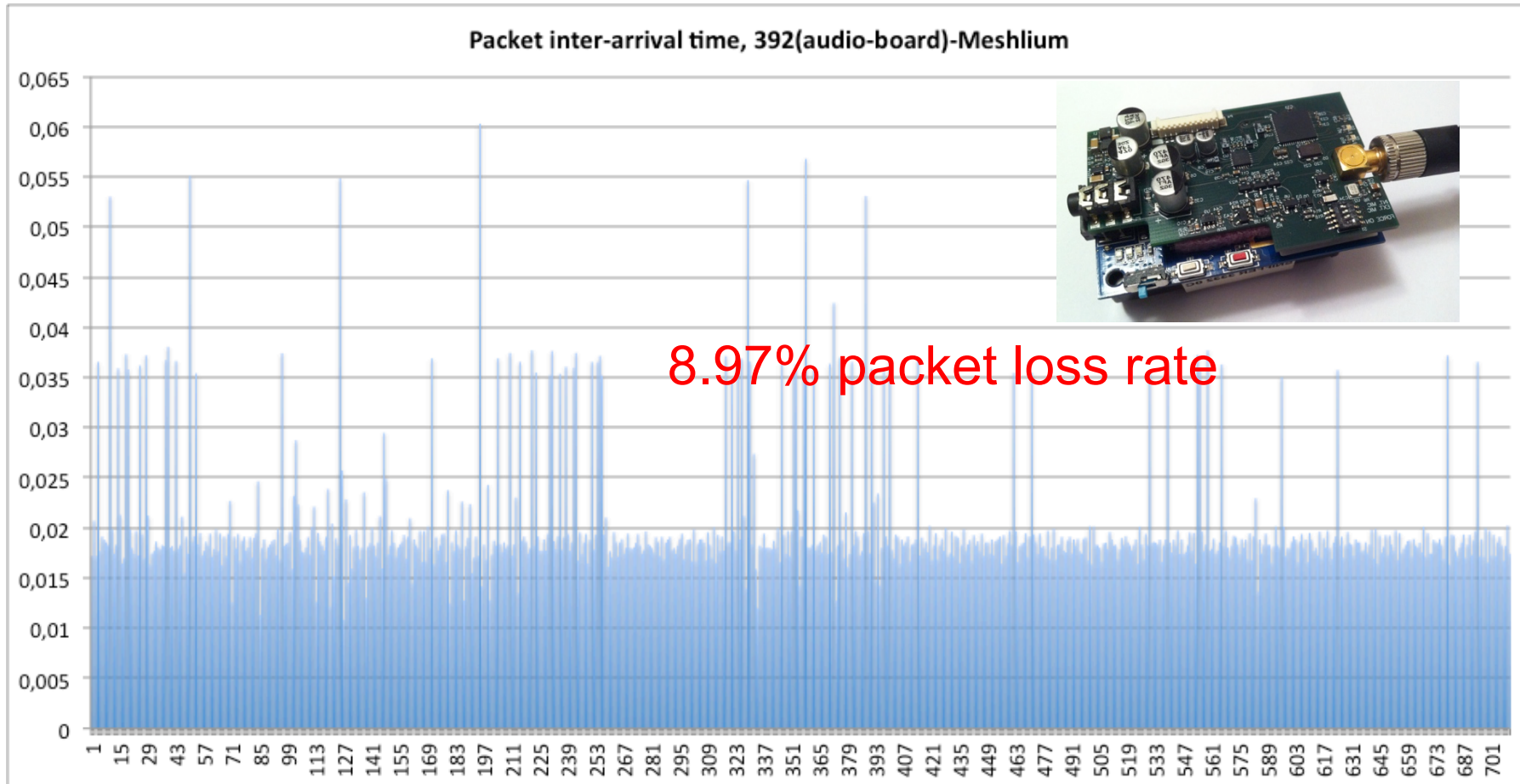
Right Screenshot: Shows a detailed view of a frame (Frame 2252) with a filter 'wpan.frame_type == 0x0001'. The packet details pane shows the IEEE 802.15.4 Data section with a destination of 0x0100 and source of 0xc823. The Frame Control Field is expanded, showing a Frame Type of 0x0001 and Security Enabled: False.

At the bottom of the interface, the status bar indicates: File: "/home/wsn/Desktop/audio_..."; Packets: 2899 Displayed: 2210 Marked: 0 Load time: 0:00.050; Profile: Default.

1-hop WaspMote audio



1-hop Advanticsys audio board

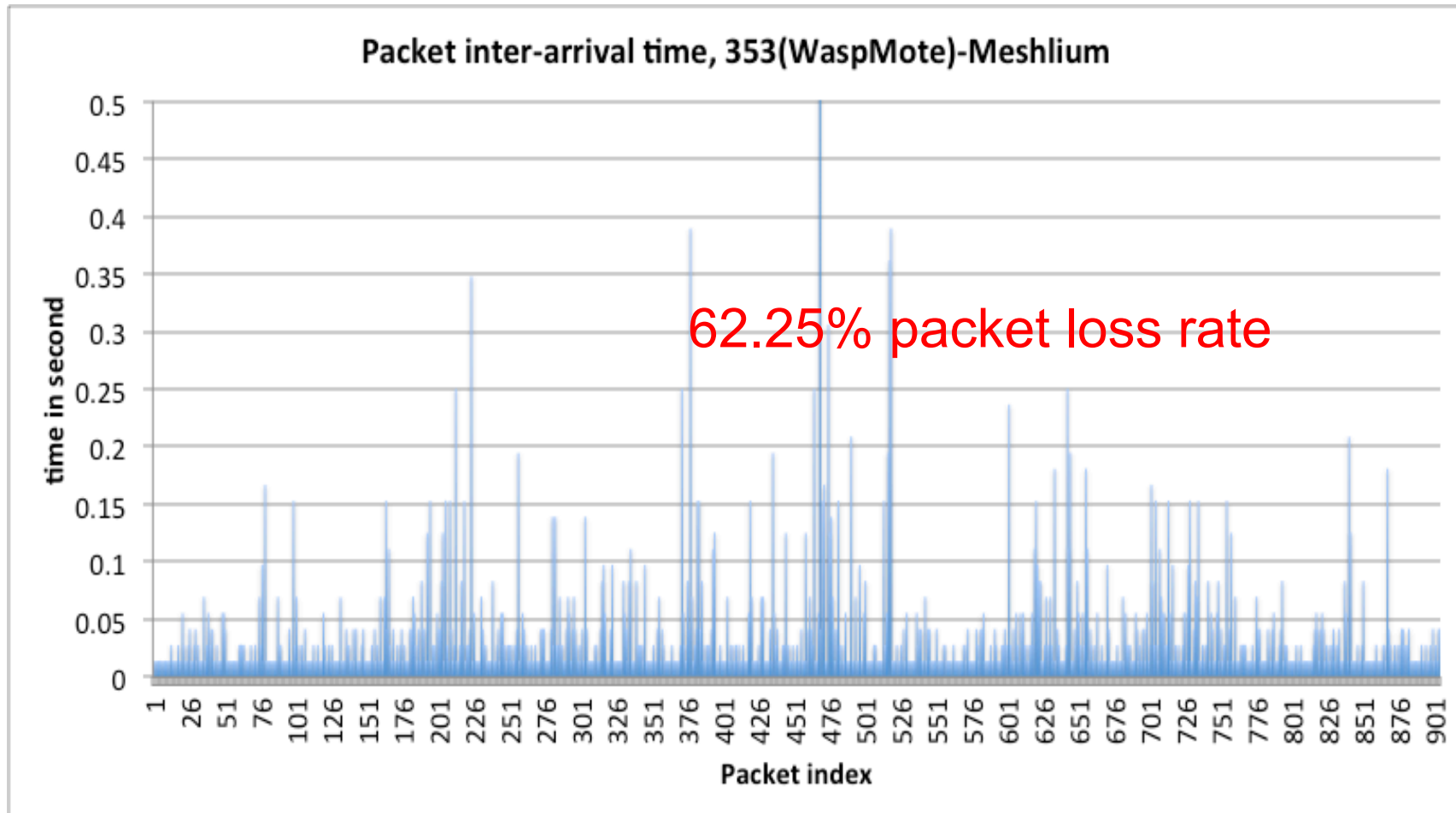


1-hop audio in non LoS

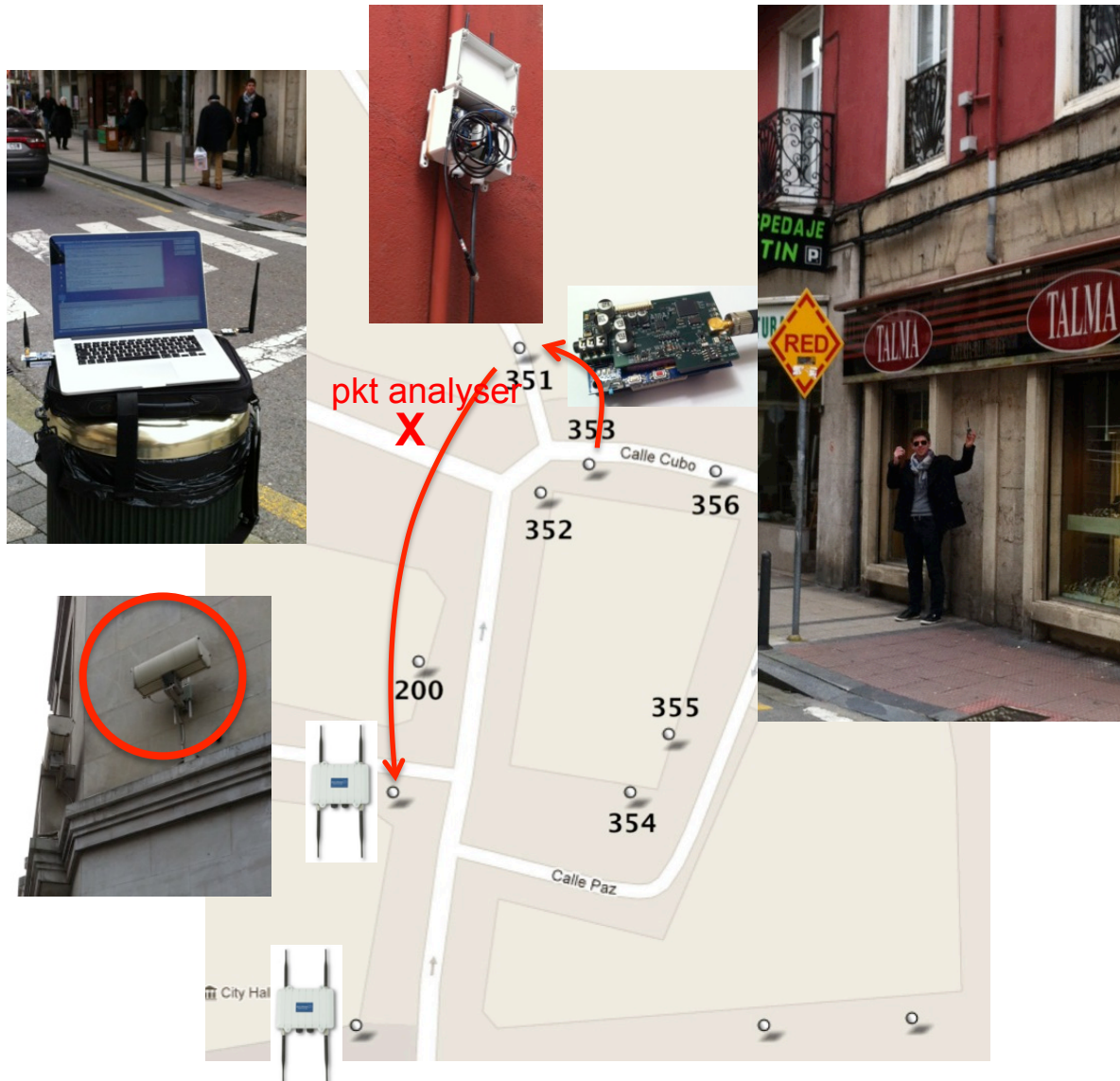
Test 7 – radio channel 18
WaspMote 8KHz
353 → Meshlium



1-hop WaspMote audio

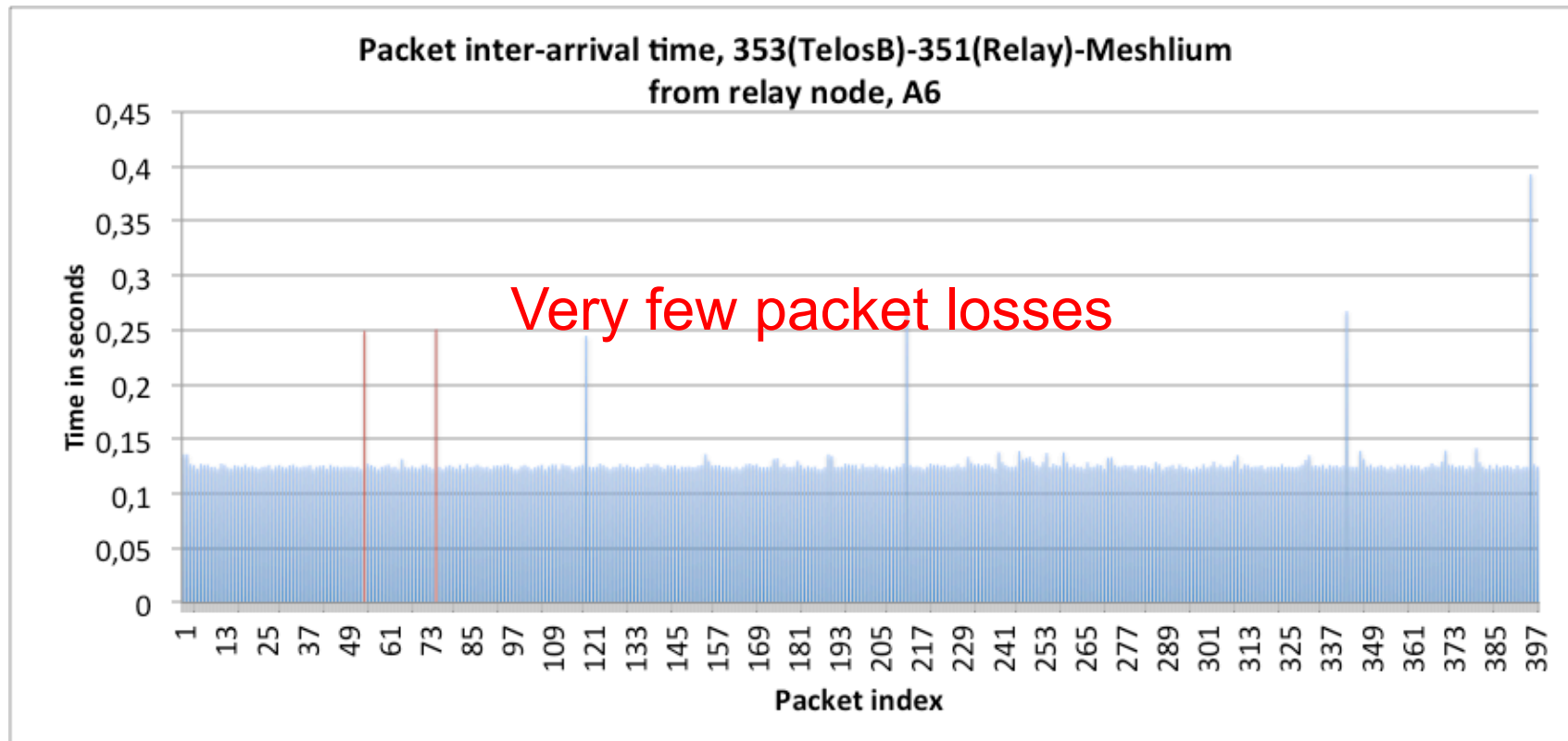


2-hop in non LOS





2-hop TelosB audio board

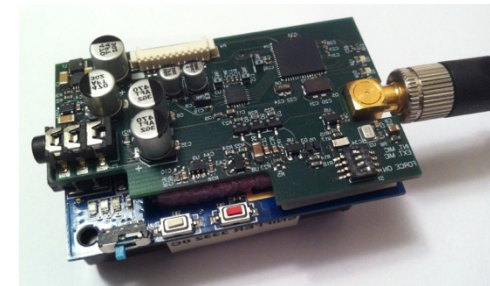
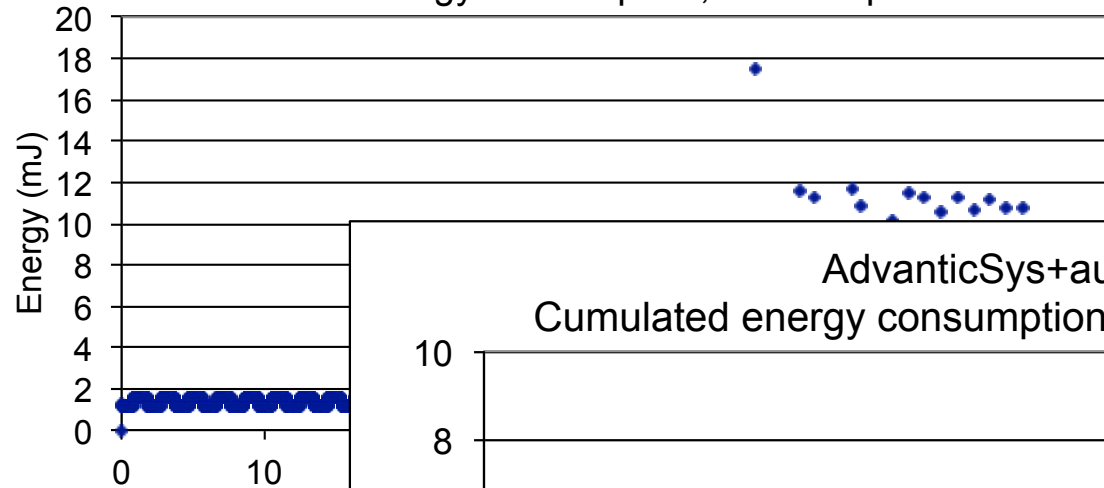


Usage of encoded audio allows for multi-hop transmission, improving greatly the transmission quality

Energy consumption

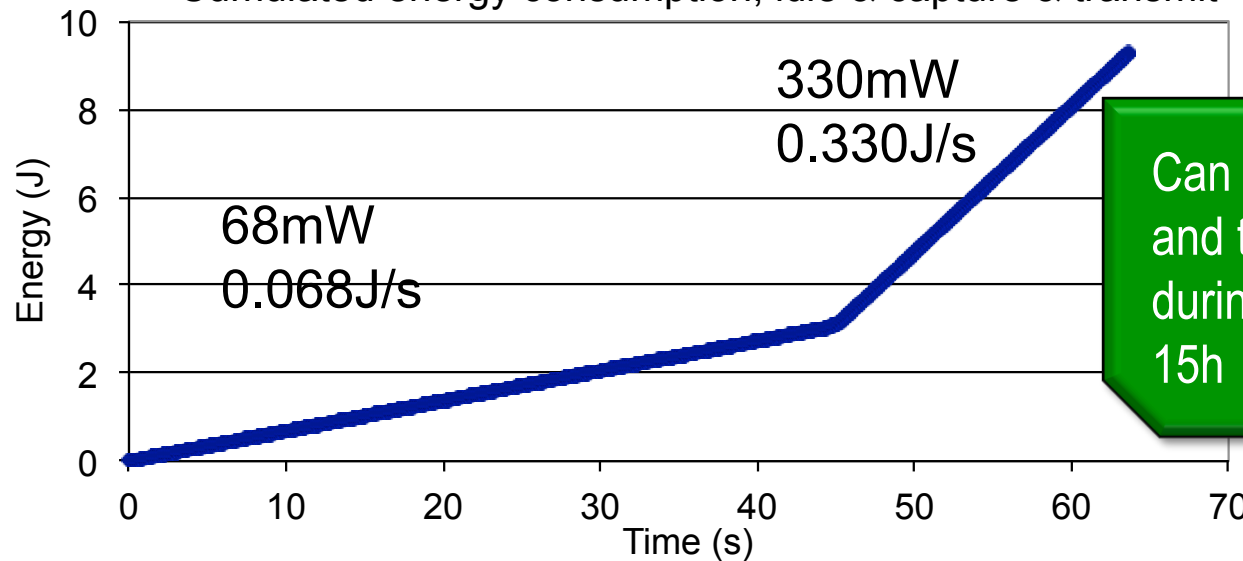
AdvanticSys+audio board

Instantaneous energy consumption, idle & capture & transmit



AdvanticSys+audio board

Cumulated energy consumption, idle & capture & transmit

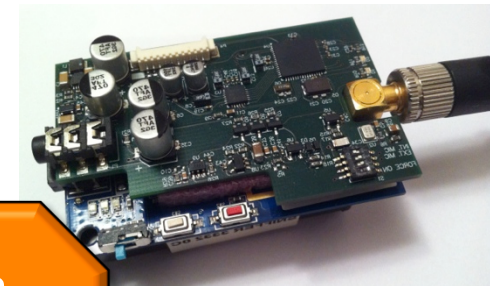
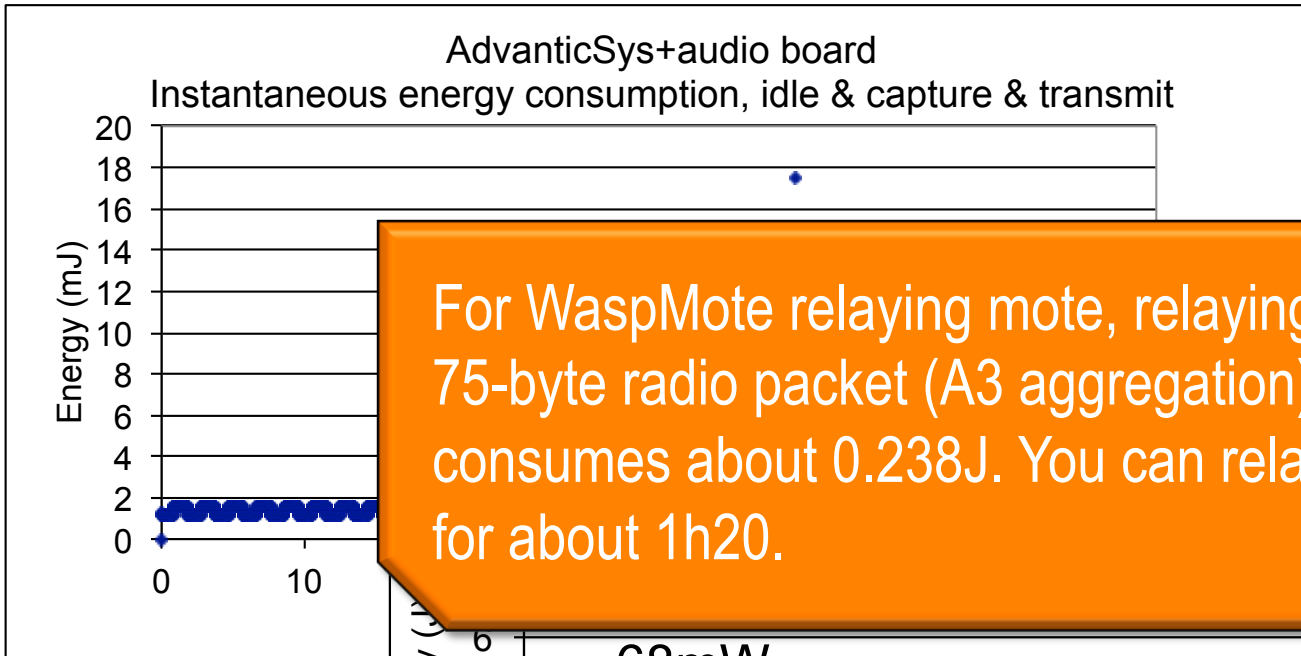


Can capture and transmit during about 15h

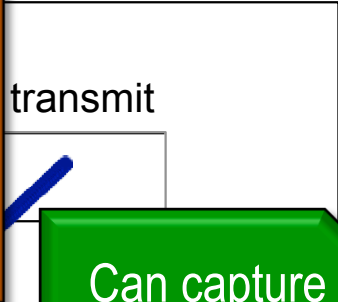


18720 JOULES

Energy consumption



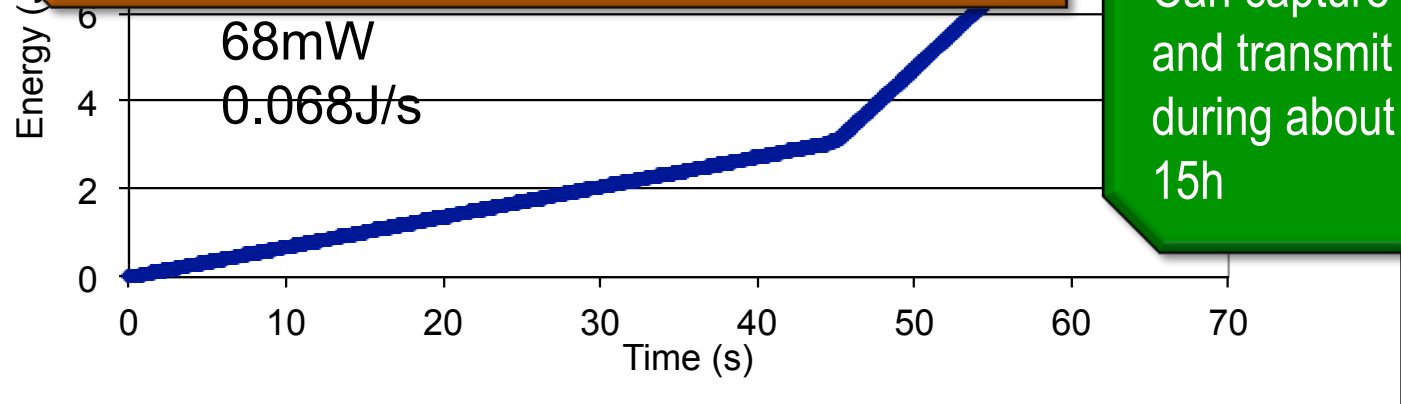
For WaspMote relaying mote, relaying a 75-byte radio packet (A3 aggregation) consumes about 0.238J. You can relay for about 1h20.



Can capture and transmit during about 15h



18720 JOULES



Conclusions

- Low-resource devices (sensor, IoT, ...) are currently deployed in a number of projects, especially in SmartCities context
- Benchmarking such test-beds is of prime importance for understanding the infrastructure limitations
- The EAR-IT project focuses on acoustic data, deployed on large scale test-beds, with very innovative applications
- We shown main performance issues as well as quality and usability indicators for streaming acoustic data
- Synthetic workload and in-situ tests have been performed to quantify the test-bed capacity and to propose adequate mechanism to provide near real-time acoustic data
- Same methodology can be applied to other test-beds, see the proposed methodology and tools:

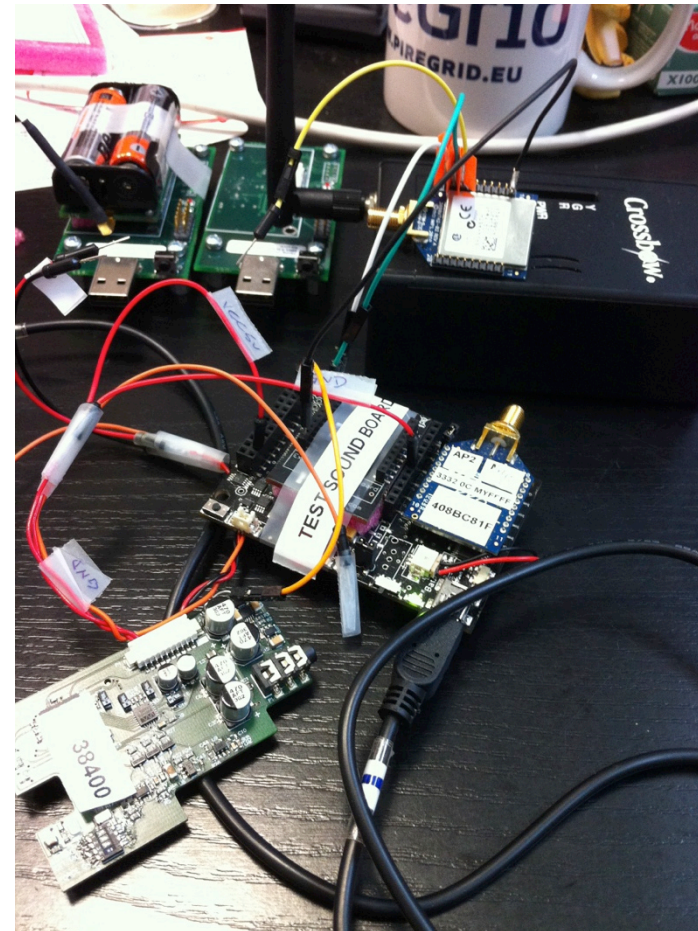
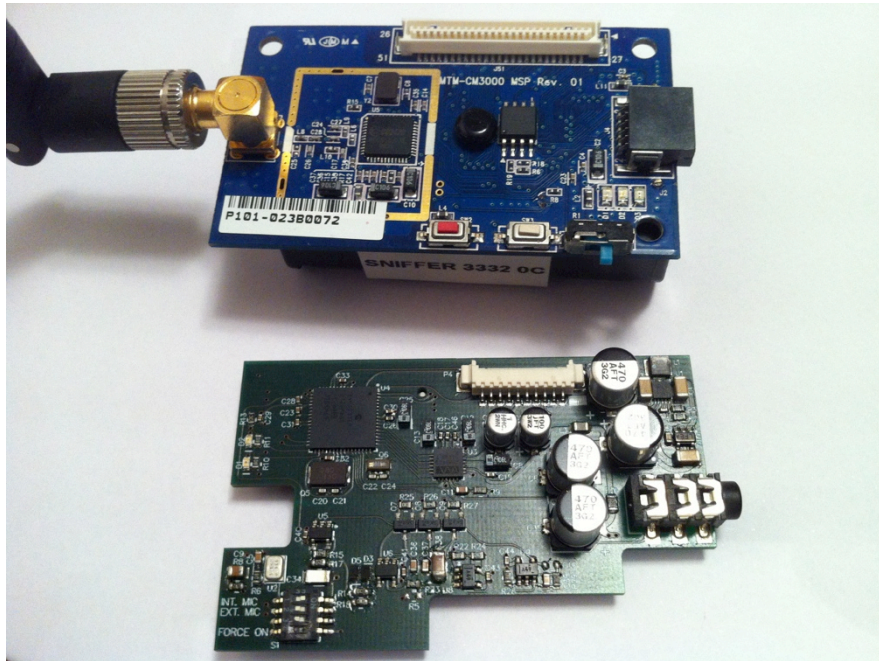
<http://www.ear-it.eu/audio-benchmarking>



Additional references

1. C. Pham, "Communication performances of IEEE 802.15.4 wireless sensor motes for data-intensive applications: a comparison of WaspMote, Arduino MEGA, TelosB, MicaZ and iMote2 for image surveillance", Journal of Network and Computer Applications (JNCA), Elsevier, 2014, DOI information: 10.1016/j.jnca.2014.08.002.
2. C. Pham, P. Cousin, A. Carer, "Real-time On-Demand Multi-Hop Audio Streaming with Low-Resource Sensor Motes", Proceedings of IEEE SenseApp, in conjunction with LCN 2014, Edmonton, Canada, September 2014.
3. C. Pham and P. Cousin, "Benchmarking low-resource device test-beds for real-time acoustic data", Proceedings of the 9th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (TridentCom'2014) , Guangzhou, China, May 5-7, 2014.
4. C. Pham, "Communication performance of low-resource sensor motes for data-intensive applications", Proceedings of the IFIP Wireless Days International Conference (WD'2013), Valencia, Spain, November 2013.
5. C. Pham and P. Cousin, "Streaming the Sound of Smart Cities: Experimentations on the SmartSantander test-bed", Proceeding of the 2013 IEEE International Conference on Internet of Things (iThings2013), Beijing, China, August 20-23, 2013.

Developped audio board



Do not hesitate to contact us



the sounds of smart environment



Thank you for your attention.
Questions ?