EPOSMote, A Internet das Coisas e os Sistemas Ciber-físicos

Arliones Hoeller Jr e Antônio Augusto Fröhlich

http://epos.lisha.ufsc.br

ESSE 2012

8 de novembro de 2012

Quem somos nós?

- LISHA
  - Laboratório de Integração Software/Hardware
  - Universidade Federal de Santa Catarina
  - Florianópolis/SC – Brasil
  - Fundado em 1985

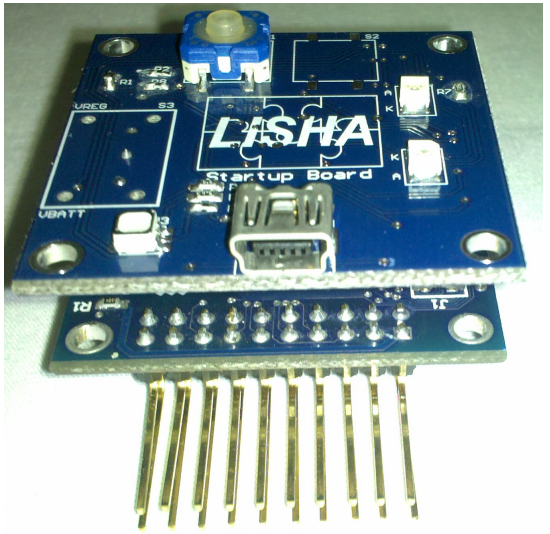
- Pesquisa
  - Sistemas Embarcados
  - Sistemas Operacionais
  - Redes de Computadores

Agenda

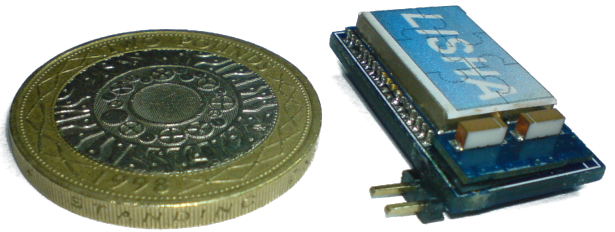
- Introdução
- Conceitos básicos de RSSF
  - Aplicações
  - Módulos de sensoriamento
  - Sensores e aquisição de dados
  - Controle de acesso ao meio (MAC)
  - Sistemas operacionais para RSSF
- Paralelo com projetos OpenEPOS e EPOSMote
- Exercícios de programação no OpenEPOS

Motivação

- OpenEPOS
  - Sistema Operacinal Livre
- EPOSMote
  - Hardware Livre!
- Por que?
  - Custos!
  - Modularidade!
- Objetivos
  - Preço: menos de R$100,00
  - Modularidade: módulos de energia alternativa

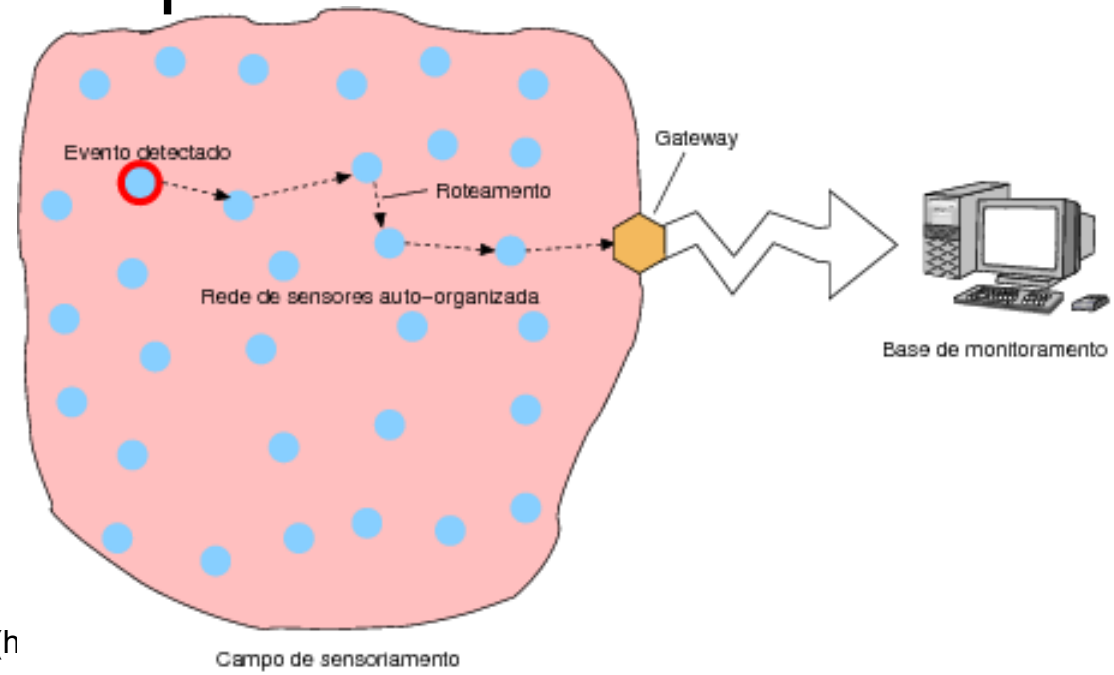


- http://epos.lisha.ufsc.br



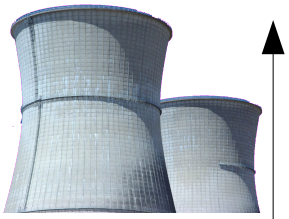
Redes de Sensores Sem Fios

- Sensoriamento pervasivo
  - Sensores complexos
  - Escala microscópica
  - Baixíssimo consumo de energia
- Plataformas de Hardware
  - Microcontrolador
  - Comunicação de baixa potência
  - Módulo de energia
  - Sensores diversos



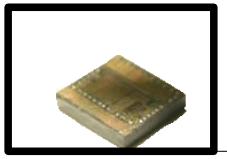
Barco Solar

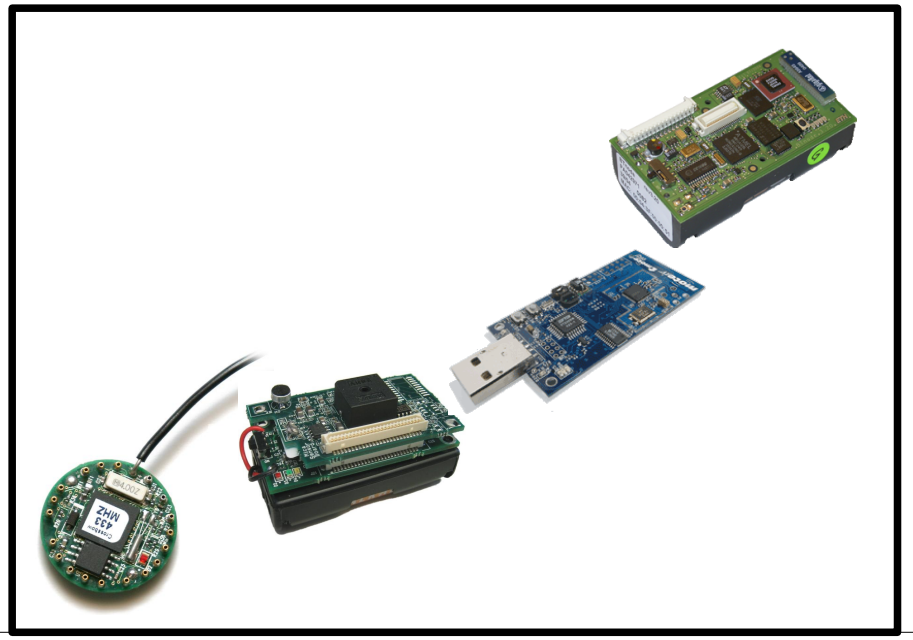
Hardware



Consumo de energia

< 10 mW



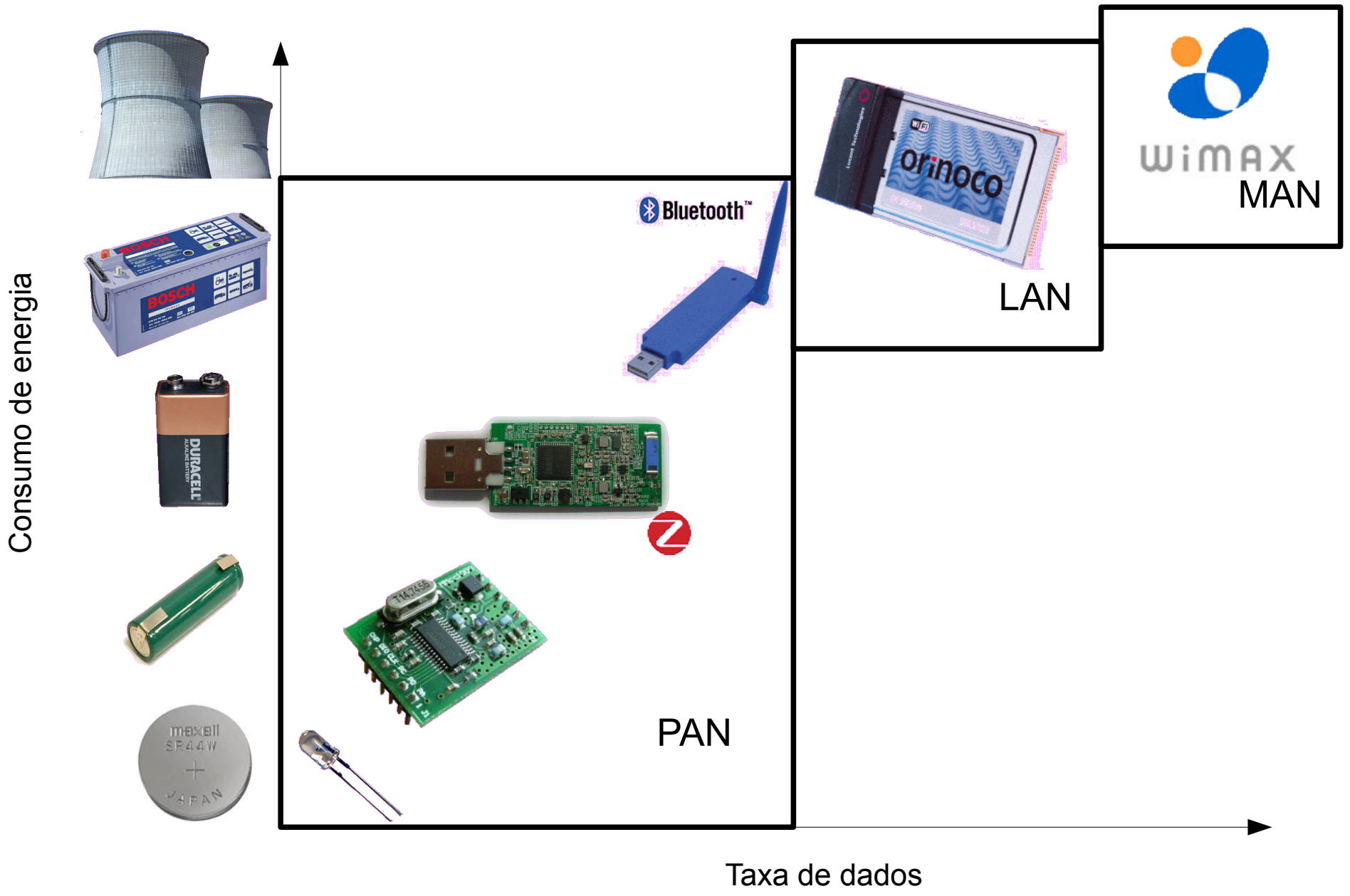


4 ~ 16 Mhz
4~128 KB RAM
< 100 mW

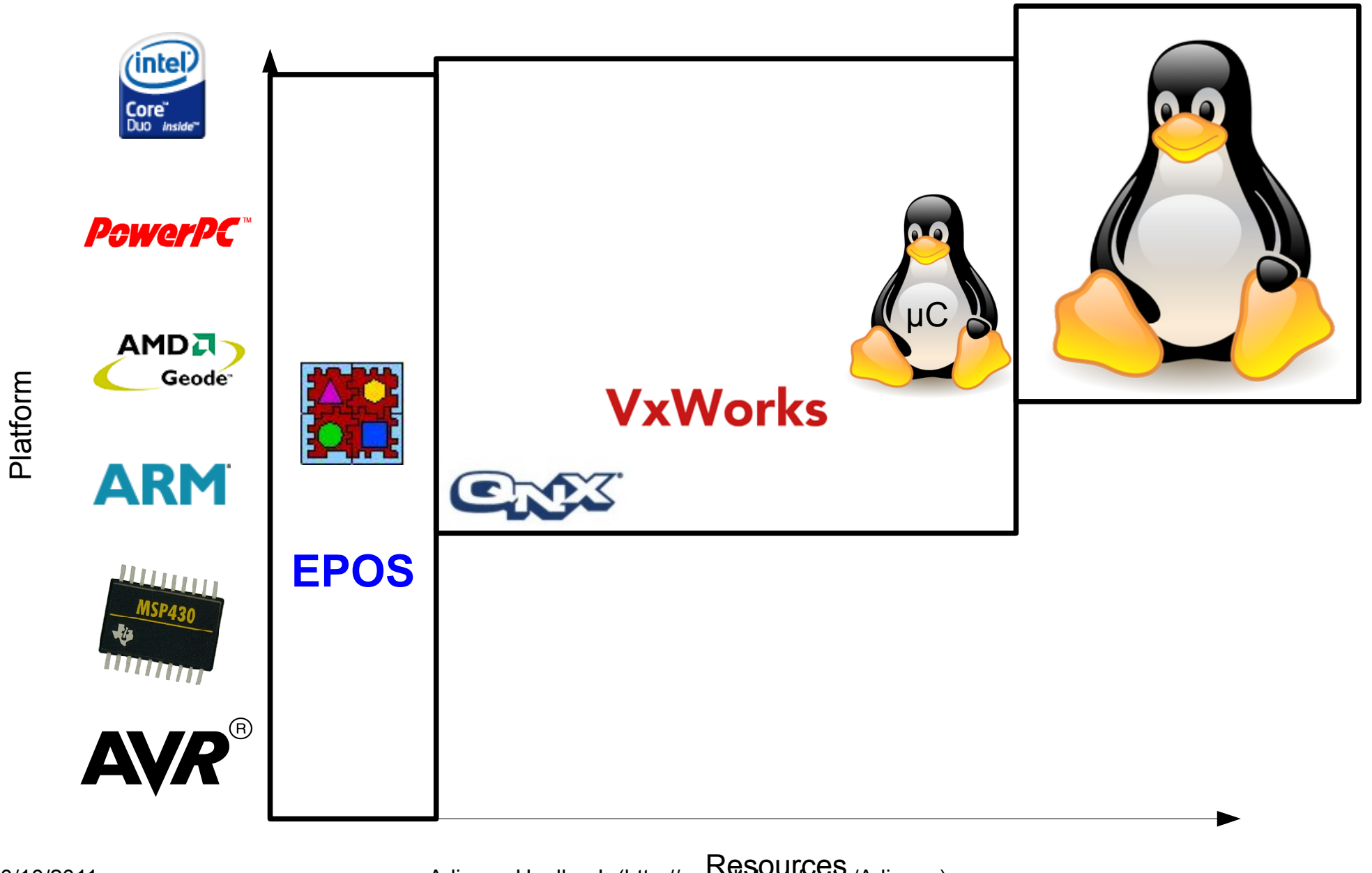
1,8 Ghz
512 MB RAM
40~80 W

Capacidade Computacional

Comunicação

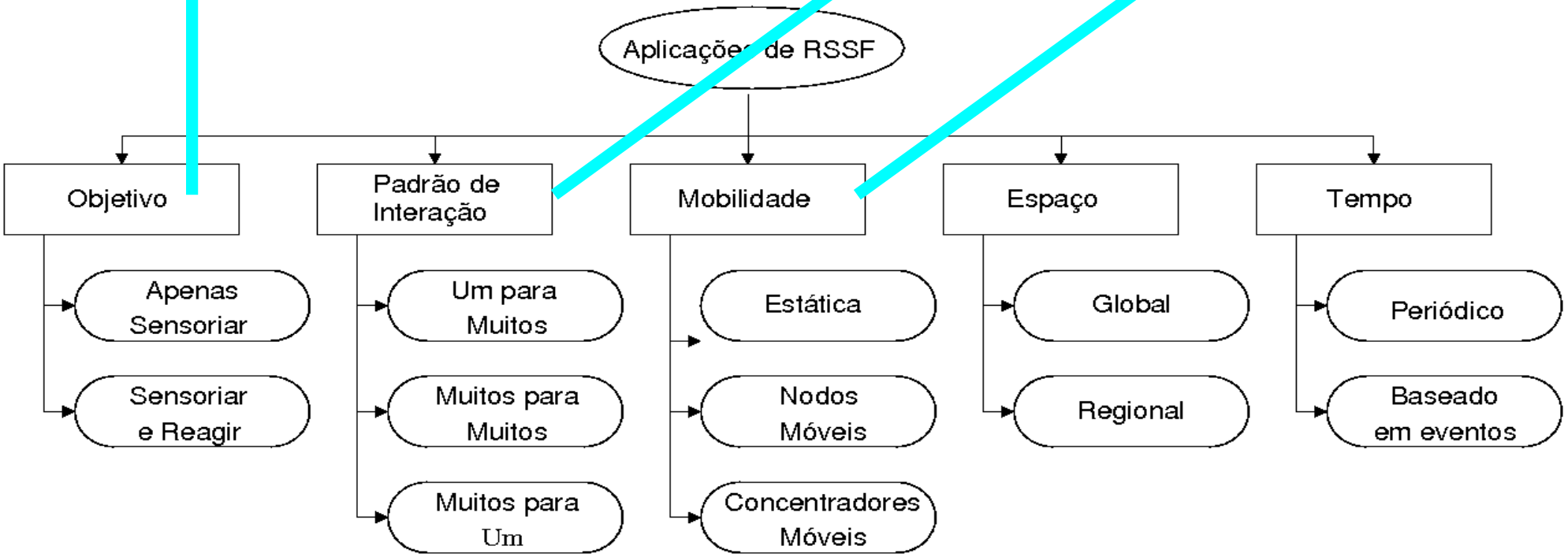


Sistemas Operacionais



Aplicações

- Plataforma depende de requisitos da aplicação
  - Demanda de processamento
  - Comunicação
    - Alcance, Taxa de dados, MAC, Roteamento
  - Heterogeneidade

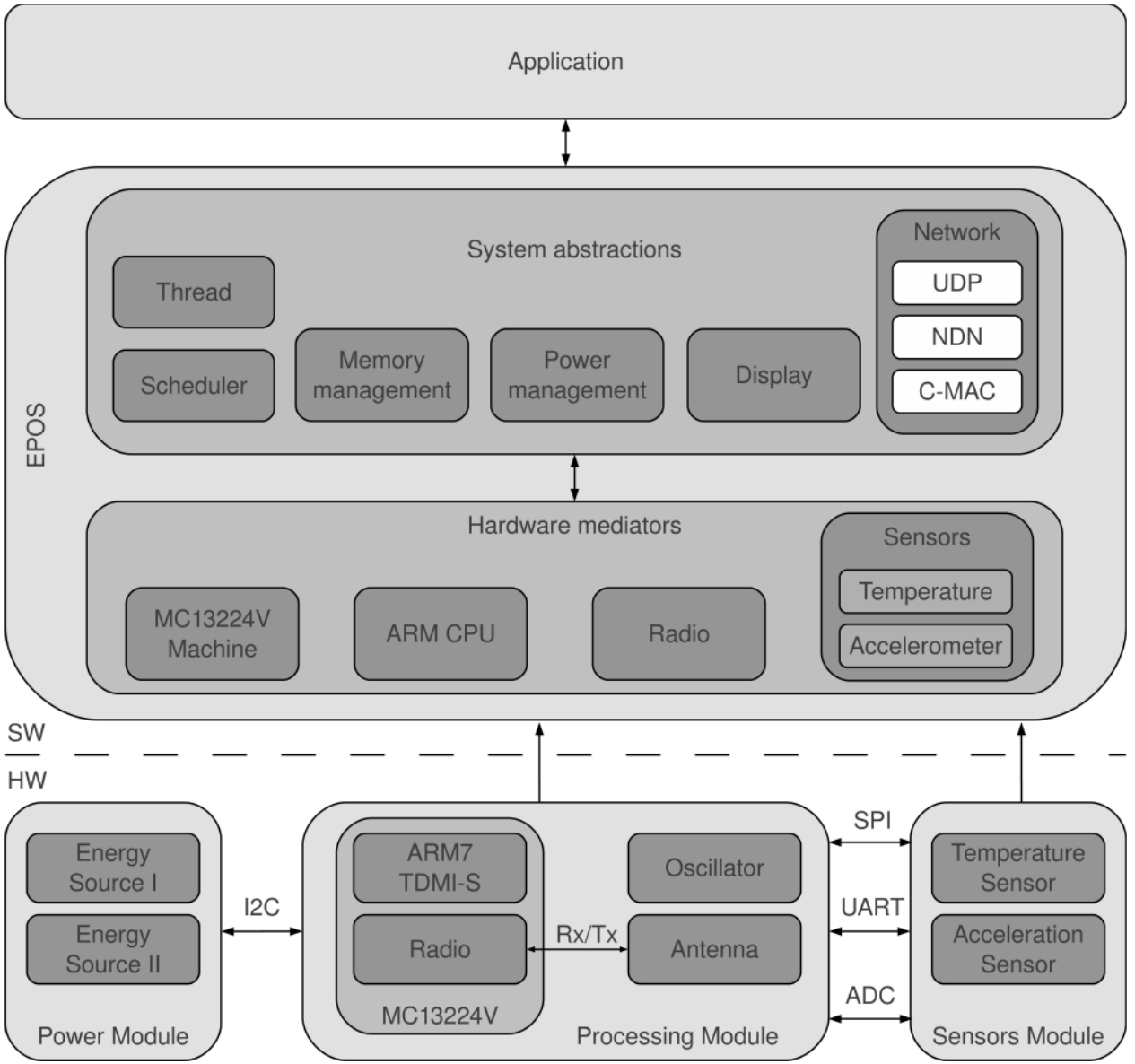


Classificação de aplicações

- Objetivo Reagir
  - + processamento
  - + energia
- Interação *-1
  - + comum
  - Sensores → Gateway
- Roteamento
  - + energia
  - + processamento
- Semântica de dados
  - Leituras individuais válidas?
- Baseado em eventos
  - - consumo
  - - interação

Aplicação	Objetivo	Interação	Mobilidade	Espaço	Tempo
Monitoramento de habitat	**AS**	***-1**	**E**	**G**	**P**
Resgate em avalanches	**AS**	***-***	**E**	**R**	**P**
Navegação de robô	**AS**	***-1**	**CM**	**R**	**BE**
Controle de tráfego veicular	**SR**	***-***	**E**	**R**	**P**

Aplicações no OpenEPOS



Aplicações no OpenEPOS

- Modelo de programação orientada a objetos
  - C++
  - Componentes com interfaces uniformes
- Biblioteca de componentes
  - Abstrações: funcionalidades alto-nível
  - Mediadores: acesso uniforme ao hardware
  - Utilitários: estruturas de dados e algoritmos
    - list, queue, random, etc
- Compilação
  - Pelo GCC, sem ferramentas complexas

Ex. 1: Aplicações no OpenEPOS

- Compilar o OpenEPOS
  - cd $EPOS && make all
- Criar arquivo para aplicação em $EPOS/app
  - Ex.: hello_epos.cc

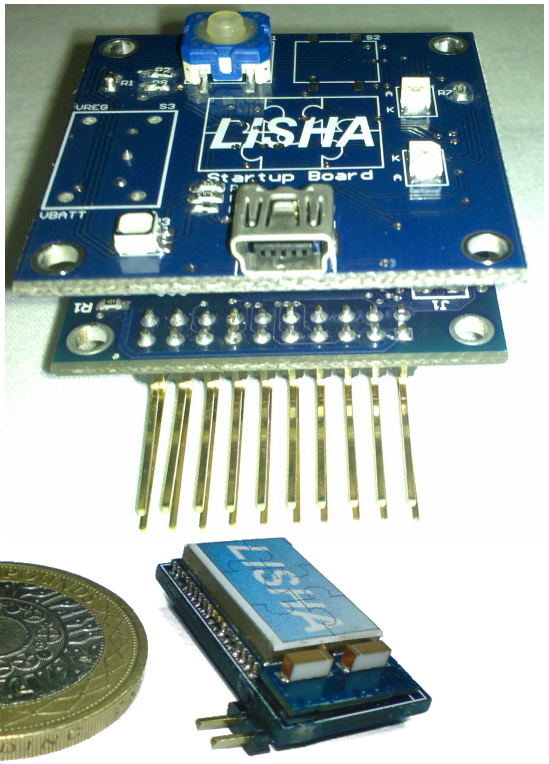
```
#include <utility/ostream.h>
__USING_SYS

int main() {
    OStream cout;
    cout << "Hello EPOS!\n";
}
```

- Compilar aplicação e gerar imagem do sistema
  - make APPLICATION=hello_epos

Módulos de sensoriamento

- Compostos por quatro componentes principais
  - processador, rádio, sensores, alimentação
- Módulos de sensoriamento devem apresentar:
  - dimensões reduzidas
  - consumo de energia reduzido
  - Modularidade
  - canal de comunicação adaptável



Requisitos de módulos (1/4)

- Dimensões reduzidas
  - Instalação não-intrusiva
  - Viabilizado pela miniaturização de componentes eletrônicos
- Grande limitação: fonte de alimentação
  - Baterias são grandes
  - Técnicas de captura de energia do ambiente são maiores ainda!
    - Painéis solares, mini-captadores eólicos, etc

Requisitos de módulos (2/4)

- Tempo de operação – duração da bateria
  - Fator determinante no projeto de hardware
  - Prioridade para componentes de baixa potência e com suporte a gerência de energia
    - DVS, DPM, sleep modes, etc
  - Decisões abrem mão de alta capacidade de processamento e potência de dispositivos

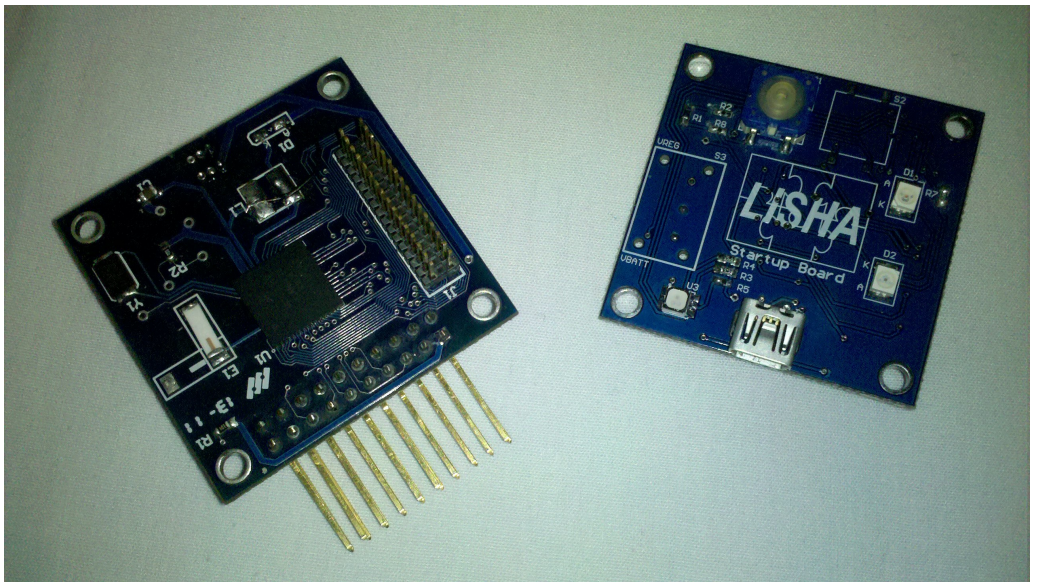
Freescale MC13224V	
Rx current	22 mA
Tx current	29 mA
Sleep current	0,85 uA
CPU	32 bits @ 24 MHz
RAM	96 KB
Flash	128 KB
Tamanho (mm)	9,5 x 9,5 x 1,2

X

MeshNetics ZigBit	
Rx current	19 mA
Tx current	18 mA
Sleep current	6 uA
CPU	8 bits @ 8 MHz
RAM	8 KB
Flash	128 KB
Tamanho (mm)	18,8 x 13,5 x 2

Requisitos de módulos (3/4)

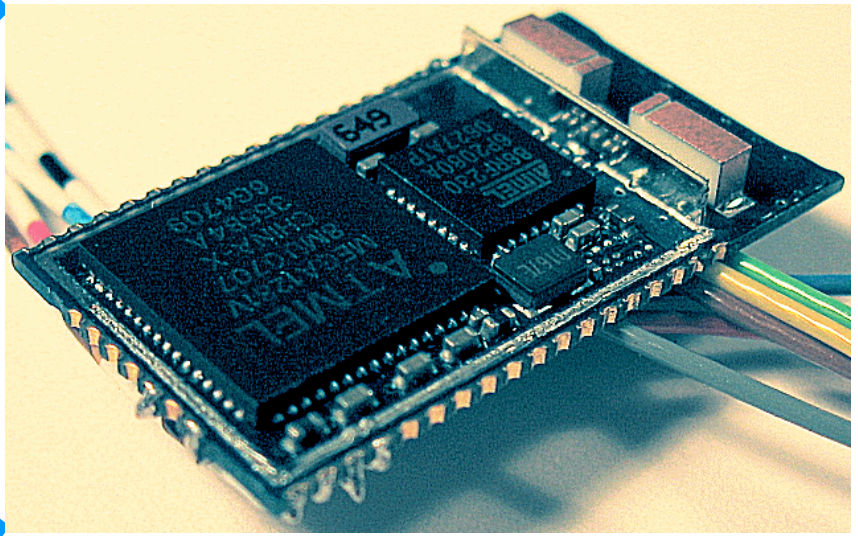
- Modularidade
  - Diferentes aplicações demandam diferentes sensores
  - Hardware não utilizado gera desperdício
    - de espaço e de energia
  - Remoção e inclusão de componentes facilitada



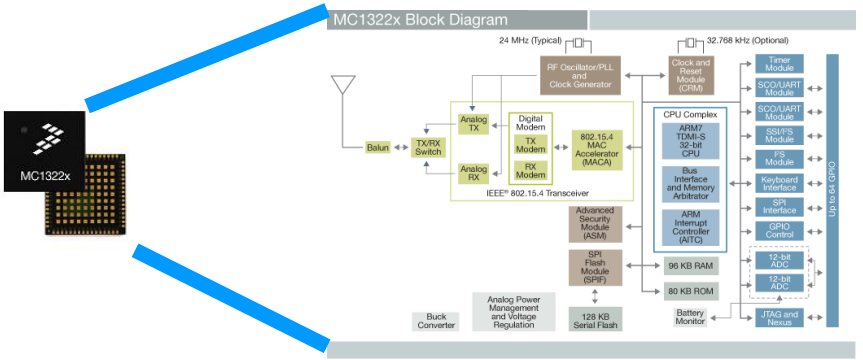
Requisitos de módulos (4/4)

- Ampla configuração do canal de comunicação
  - Transceptor é o vilão do consumo de energia
  - Melhor permanecer desligado sempre que possível
  - Aplicações
    - padrões de comunicação específicos
    - beneficiam-se de características do canal de comunicação
      - modulação
      - MAC
    - permite controle do consumo de energia sem comprometer comunicação

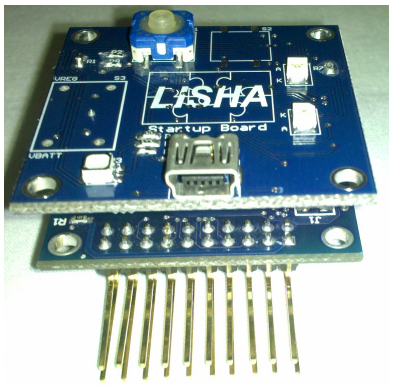
Integração de componentes



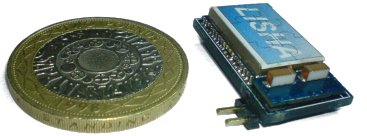
single-package



single-die (SoC)



EPOSMote II



EPOSMote I

Comparação dos módulos

Módulo	MicaZ	ZigBit	MC13224V
Encapsulamento	módulo	single-package	single-die (SoC)
Processador	8-bits ATMega128L	8-bits ATMega1281v	32-bits ARM7 TDMI
RAM	4 kB	8 kB	96 kB
Flash	128 kB	128 kB	128 kB
Potência de transmissão	0 dBm	+3 dBm	+4 dBm
Corrente Tx*	17.4 mA	18 mA	29 mA
Corrente Rx*	19.7 mA	19 mA	22 mA
Corrente *sleep*	15 uA	6 uA	0.85 uA
Dimensões (mm)	58 x 32 x 7	24 x 13,5 x 2	9,5 x 9.5 x 1,2
Preço (USD)	$ 144,00 (2005)	$ 18,25	$ 4,50

* Corrente drenada quando operando na potência máxima de transmissão ou recepção.

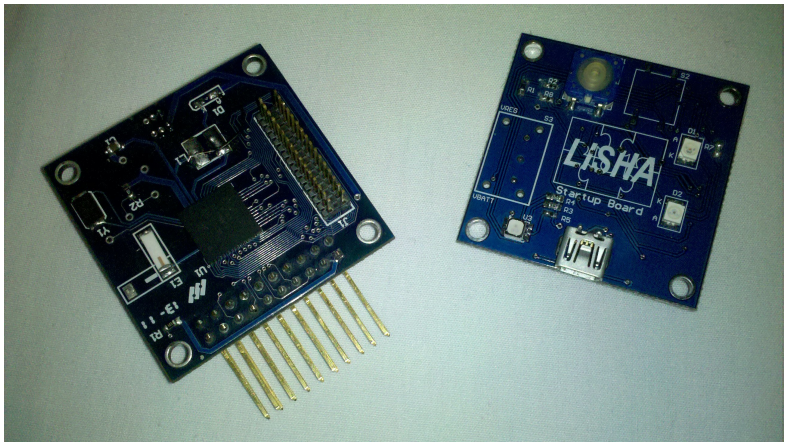
EPOSMote

- Desenvolvido na UFSC – LISHA
- Objetivos
  - Desenvolver uma família de motes modulares, configuráveis, de custo acessível e que atenda às necessidades do LISHA
  - Servir de plataforma para validação do OpenEPOS
    - serviços de SO para RSSF
    - C-MAC: Configurable MAC
    - HECOPS: localização baseada em RSSI
    - Roteamento: ACO + ZRP
    - Fontes alternativas de energia (energy harvesting)

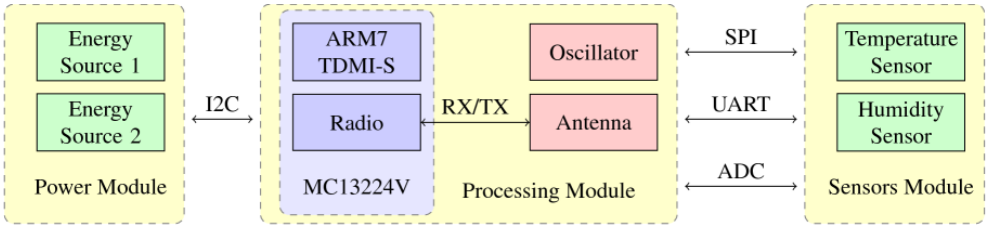
EPOSMote I

- MeshNetics ZigBit
  - 8-bits ATMega1281v
    - 128 KB Flash + 8 KB RAM
  - AT86RF230
    - -101dBm sensibilidade Rx
    - +3dBm potência Tx
    - IEEE 802.15.4
    - 2.4 Ghz ISM
- Sensor SHT-11
  - Integrado: temperatura e humidade
  - Digital: SPI

EPOSMote II



- Projeto modular
  - Interfaces padrão
  - 3 módulos:
    - Base: CPU + rádio
    - I/O: sensores
    - Alimentação: bateria ++
- Freescale MC13224v
  - 32-bits ARM7 TDMI
    - 128 KB Flash + 96 KB RAM
  - Rádio integrado
    - -96 dBm sensibilidade Rx
    - +4 dBm potência Tx
    - IEEE 802.15.4
    - 2.4 Ghz ISM



EPOS Mote II block diagram

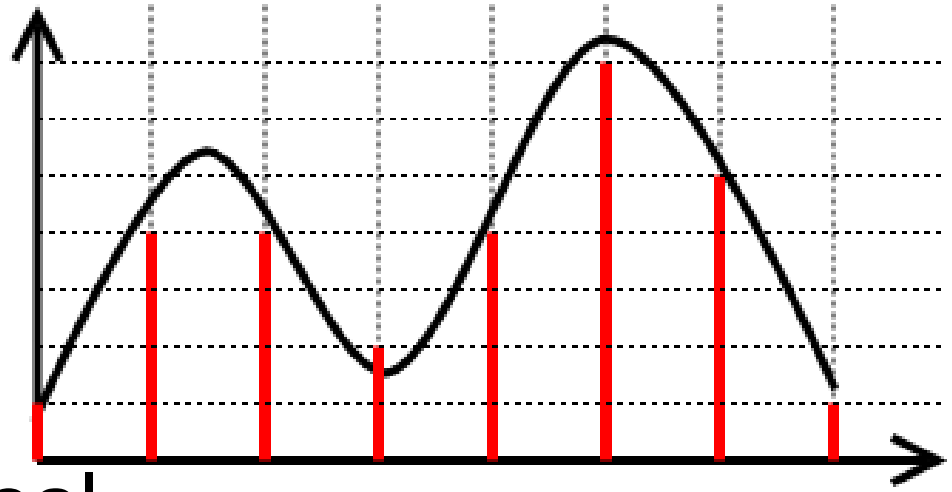
Sensores e aquisição de dados

- Sensor
  - responde estímulos físicos
  - luz, temperatura, etc
  - transmite impulso resultante
  - corrente, tensão, etc

- Normalmente interage com sistema digital
  - através de um ADC (Analog-to-Digital Converter)
  - provendo informações sobre o mundo analógico

Sinais: interface com o mundo real

- O mundo é analógico
- Computadores digitais são... digitais (e geralmente binários)!
- Permitir que um processador digital interprete sinais analógicos é (obviamente) importante
  - Áudio e vídeo digitais
  - Telefonia digital
  - Sensores e atuadores
- Conversão
  - Amostras periódicas
  - Aproximação do valor real

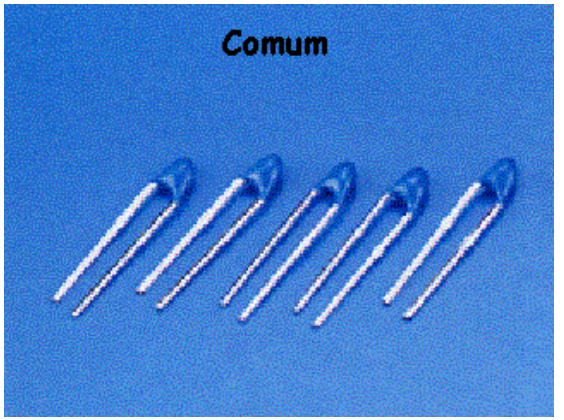


Tipos de sensores

- Analógicos: sistema digital deve ter um ADC
  - ADC!
  - Variam resistência dum material ou corrente num circuito
  - Ex.: termistores, foto-diodos, etc
- Digitais: ADC integrado ao sensor, dados enviados digitalmente ao sistema
  - ON/OFF: chaves, boias, etc
  - SPI
  - RS-232 / 485 (UART)
  - I2C

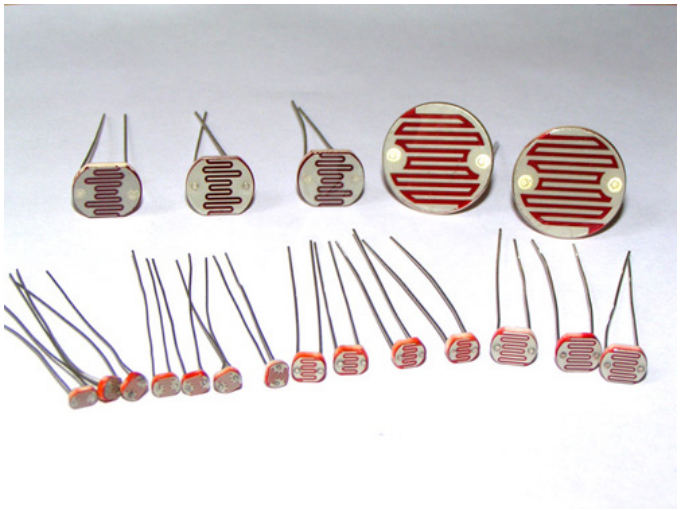
Exemplos de sensores (1/4)

- Temperatura
  - Termistores: analógicos
    - Curva de resistência (Steinhart-Hart)
    - ADC/circuito divisor de tensão
  - SHT-11: digital
    - Auto-calibrado
    - SPI
    - Temperatura + humidade



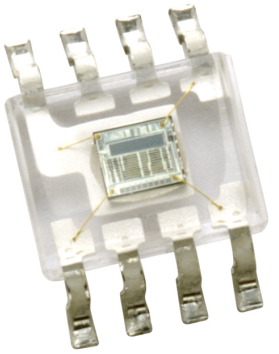
Exemplos de sensores (2/4)

- Luz
  - Foto-resistores e Foto-diodos: analógicos
    - Curvas de resistência ou corrente
    - Ampla gama de frequencias (IR, visível, UV)
    - ADC com divisor de tensão ou circuito RC auxiliar
    - Diodos são mais precisos



Exemplos de sensores (3/4)

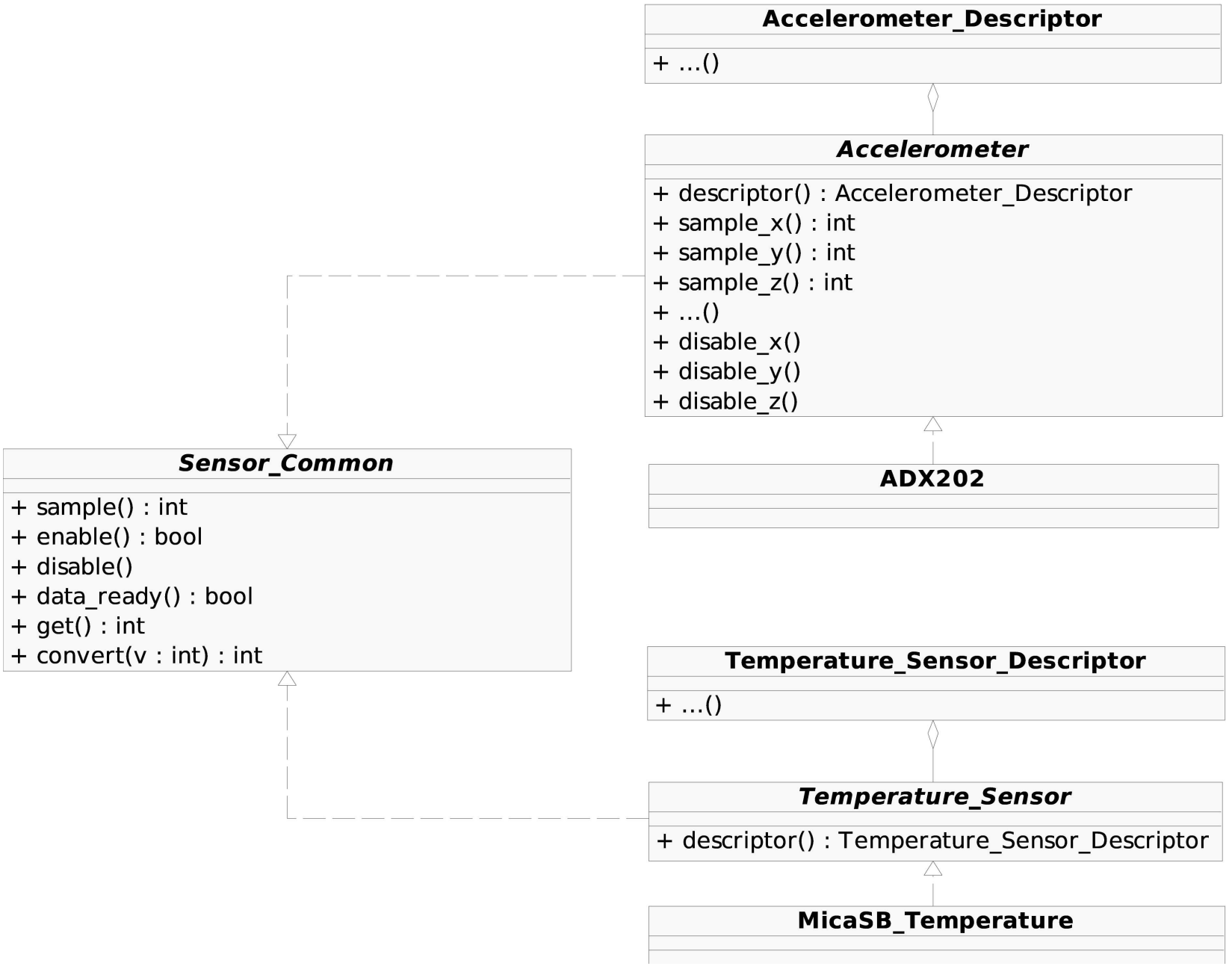
- Luz
  - TAOS TSL2550: digital
    - Dois foto-diodos
      - Um sensível a IR e luz visível
      - Outro sensível primariamente a IR
      - Senbilidade equiparada à do olho humano
    - ADC de 12 bits
    - SMBus (2 fios)
    - Medições em Lux



Exemplos de sensores (4/4)

- Campo magnético
  - Honeywell HMC100x: analógicos
    - Magnetoresistivos
    - Saída linear ao campo magnético aplicado
- Aceleração
  - ADXL345BCCZ-RL7: digital ou analógico
    - SPI
    - Saídas lineares à aceleração aplicada

Abstrações de sensores no EPOS



Ex. 2: Sensores no OpenEPOS

- Criar arquivo para aplicação em $EPOS/app
  - Ex.: sensor.cc

```
#include <utility/ostream.h>
#include <sensor.h>
__USING_SYS

int main() {
  OStream cout;
  Temperature_Sensor sensor;
  cout << "Temperature = "
       << sensor.sample()
       << "\n";
}
```

- Compilar aplicação e gerar imagem do sistema
  - make APPLICATION=sensor

MAC: controle de acesso ao meio

- Numa RSSF múltiplos nós podem iniciar transmissões simultaneamente
- Meio físico compartilhado => colisão!
- Protocolo que permita múltiplo acesso necessário
- MAC: *Medium Access Control*
  - Algoritmo distribuído que controla o uso do meio físico

Abordagens para MAC

- Particionamento de canal: FDMA, TDMA
  - Divide canal disponível
  - Aloca cada divisão para uso exclusivo por um determinado elemento
- Acesso aleatório: CSMA-CA/CD, ALOHA
  - Há colisões
  - Dependendo do protocolo é possível identificar, recuperar, evitar colisões
- Passagem de permissão: Token-Bus/Ring
  - Coordenação estrita do acesso ao canal
  - Evita colisões
- Abordagens híbridas

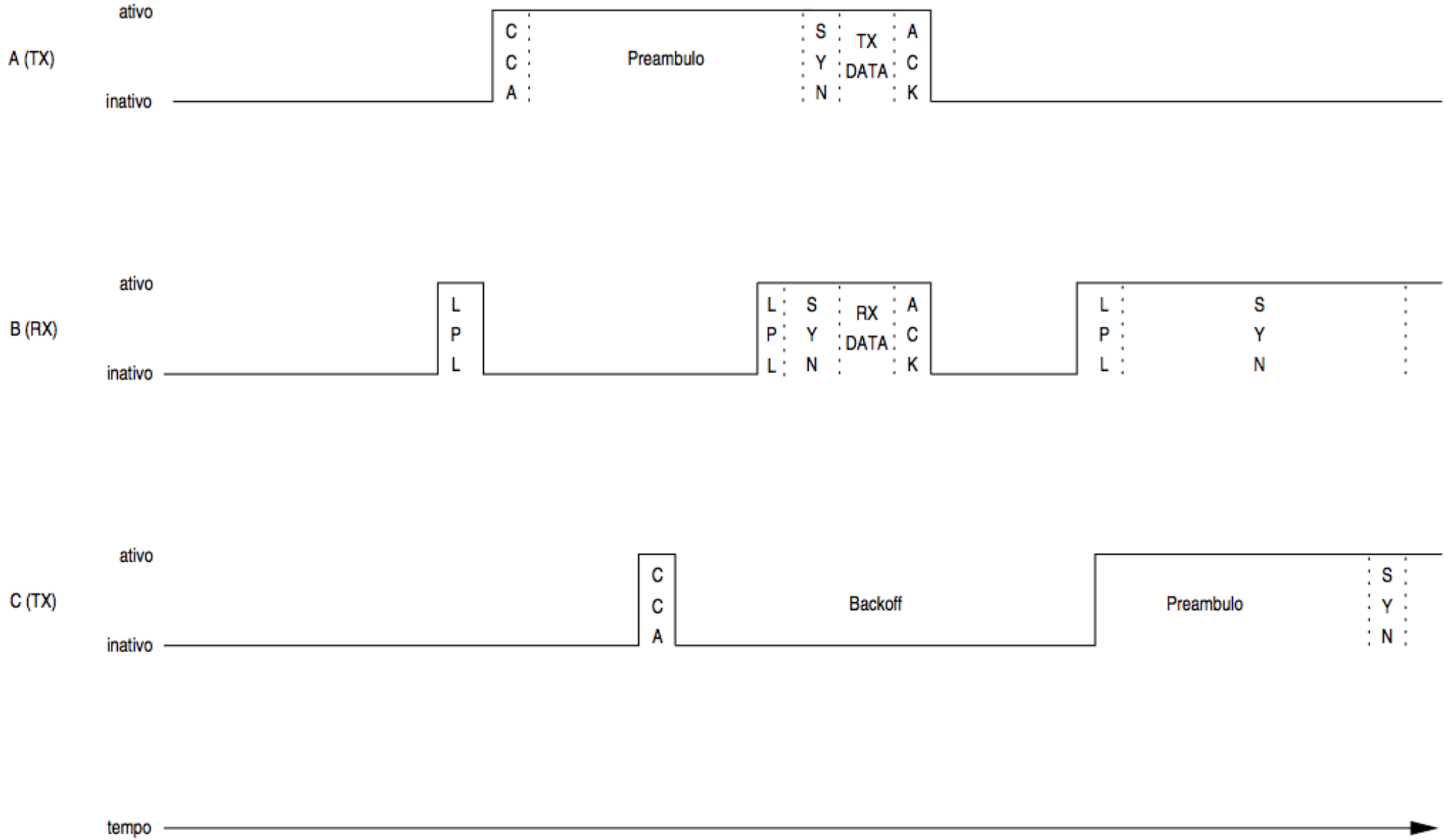
MACs para RSSF

- Módulos de RSSF têm recursos limitados
  - Baixa capacidade de processamento
  - Baixa potência de transmissão
  - Fonte limitada de energia!
- MACs precisam
  - Utilizar rádio de modo eficiente
  - Minimizar gasto de energia
- Para reduzir gasto de energia compromete-se
  - Latência
  - Vazão
  - Disponibilidade

MACs para RSSF

- Fontes de sobrecusto de energia [Langendoen and Halkes 2005]
  - Colisões: causam retransmissões
  - Sobrecusto de controle: pacotes de controle não carregam informação útil
  - Escuta ociosa (*idle listening*): ouvir por pacotes que não foram enviados
  - *Overhearing*: receber/ouvir pacotes que não são destinados ao nodo
- Propostas de MAC:
  - B-MAC, X-MAC, S-MAC, T-MAC, Z-MAC
  - IEEE 802.15.4

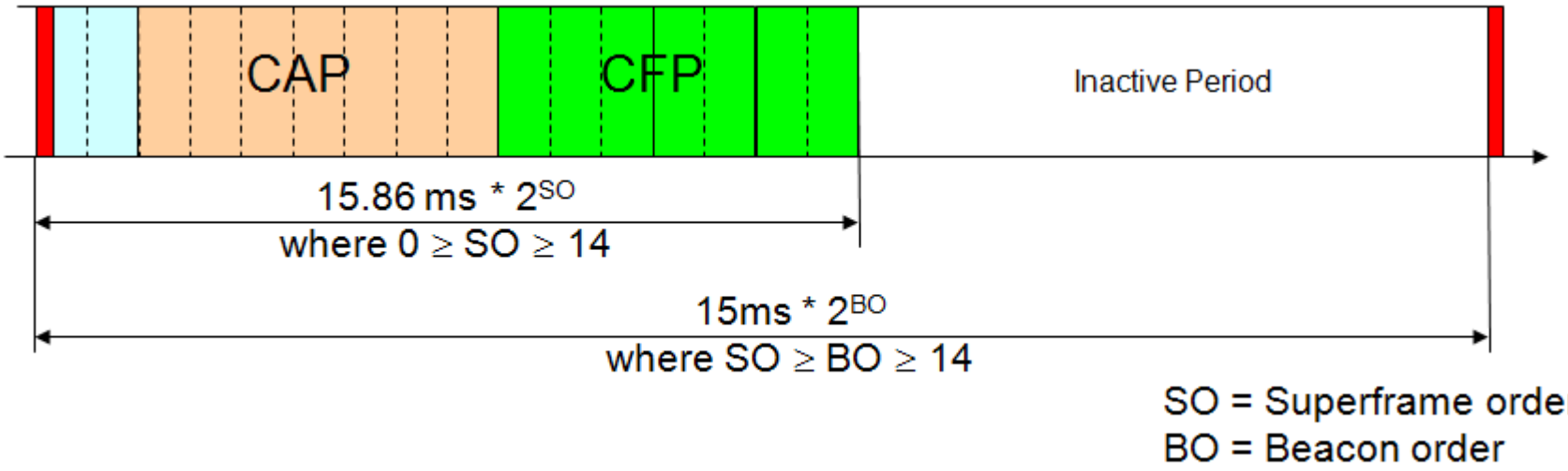
B-MAC [Polastre et al. 2004]



Z-MAC [Rhee et al. 2008]

- Híbrido: TDMA + CSMA
- TDMA como mecanismo primário
- Nó “dono” deve iniciar transmissão no início de sua fatia de tempo
- Se “dono” não utiliza a fatia, outros nós disputam o meio com CSMA
- Limitação:
  - Alocação estática das fatias de tempo

IEEE 802.15.4 [IEEE CS 2006]

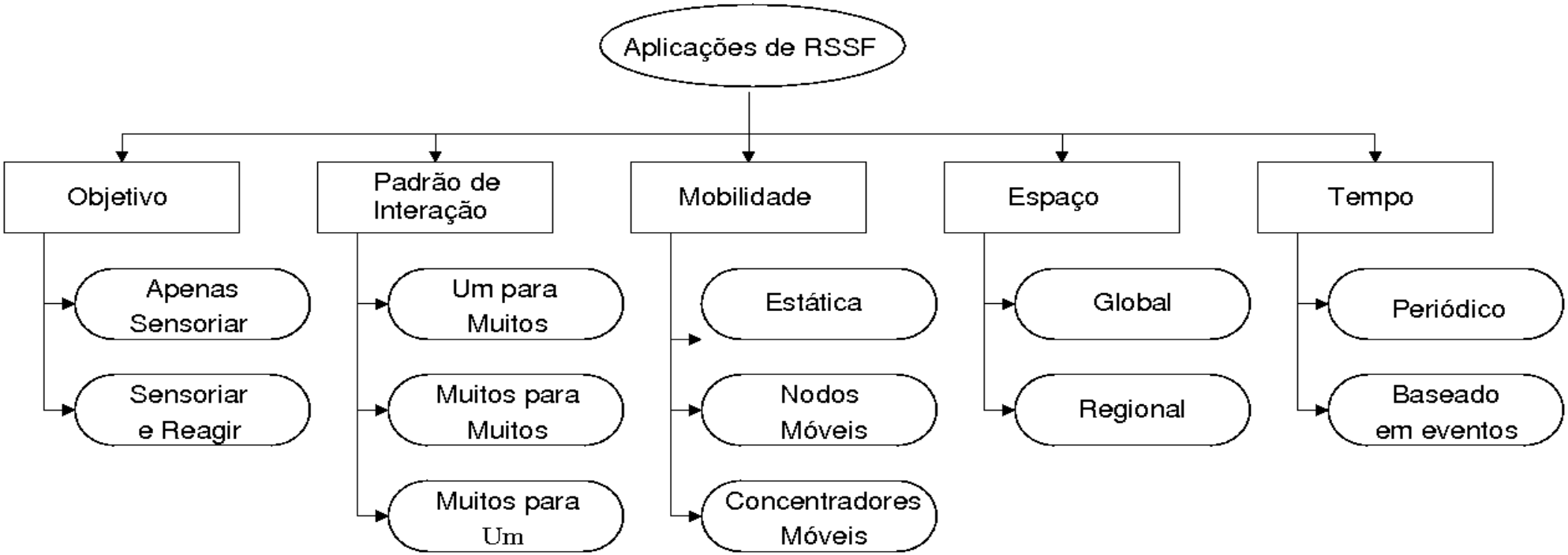


Marco Naeve, Eaton Corp.

- 2 períodos distintos: CAP e CFP
- CAP: Contention Access Period
  - CSMA/CA (RTS/CTS) + ACK
- CFP: Contention Free Period
  - *Beaconing*
  - GTS: *Guaranteed Time Slots*

Qual MAC utilizar?

- Aplicações distintas => diferentes requisitos



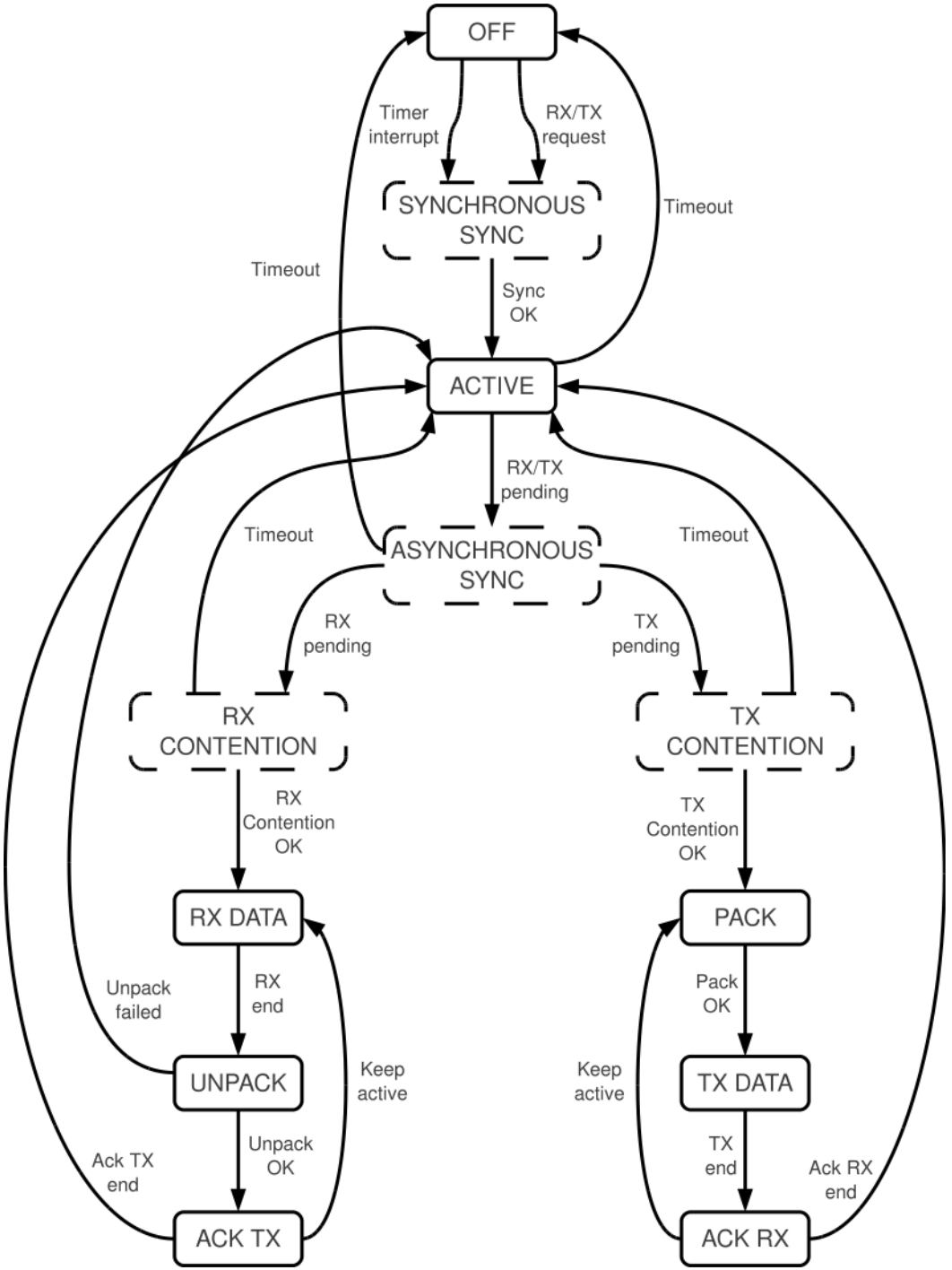
Taxonomia para aplicações de RSSF [Mottola and Picco 2010]

Qual MAC utilizar?

- Requisitos distintos => diferentes protocolos
- Categorias:
  - Verificação do meio (*channel pooling*)
  - *Scheduled contention*
  - Baseados em TDMA
  - Híbrido

- Configurable Medium Access Control Protocol
  - C-MAC
  - Protocolos específicos para aplicação
  - Configurabilidade sem prejudicar desempenho
  - Máquina de estados generalizada com funções específicas abstraídas

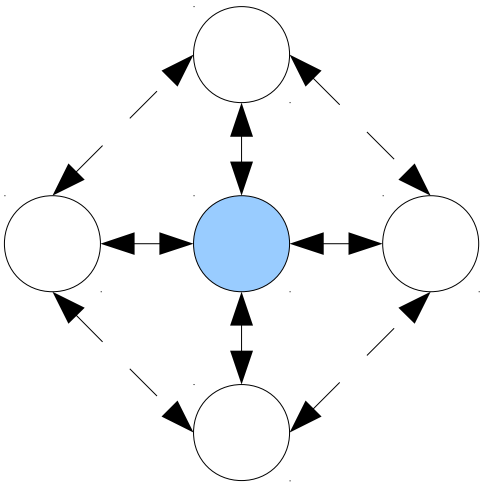
C-MAC


OFF
Timer interrupt
RX/TX request
SYNCHRONOUS SYNC
Timeout
Timeout
Sync OK
ACTIVE
RX/TX pending
Timeout
ASYNCHRONOUS SYNC
Timeout
RX pending
TX pending
RX CONTENTION
TX CONTENTION
RX Contention OK
TX Contention OK
RX DATA
PACK
Unpack failed
RX end
Pack OK
Keep active
UNPACK
Keep active
TX DATA
Ack TX end
Unpack OK
TX end
Ack RX end
ACK TX
ACK RX

C-MAC: experimentos

- Parâmetros de configuração e topologia de rede

Microcontroller clock	**1 MHz**
Packet size	**64 bytes**
Tx power	**3 dBm**
Beacon order	**7**
Superframe order	**4**
Duty cycle	**12%**



C-MAC: resultados

- Uso de memória

Configuration	Code (bytes)	Data (bytes)
No CSMA-CA / ACK	3248	185
No ACK	3572	185
No CSMA-CA	3768	202
CSMA-CA / ACK enabled	4092	202
CSMA-CA / ACK beacons enabled	5344	215

Configuration	Code (bytes)	Data (bytes)	RTT (ms)
C-MAC IEEE 802.15.4	4092	202	79
ZigBeeNet IEEE 802.15.4	26776	289	62

C-MAC: conclusões

- C-MAC está operacional
  - Configuração não é automatizada, mas é fácil
  - Configurabilidade permitida através de técnicas de metaprogramação estática

- Quão difícil é implementar um novo MAC?
  - Primeiro, o desenvolvedor necessita aprender como utilizar o framework
  - Daí, isto deve ser bastante direto

Ex. 3: Rádio no OpenEPOS

- Criar arquivo para aplicação em $EPOS/app
  - Ex.: radio.cc

```
#include <nic.h>
__USING_SYS

int main() {
  NIC * nic = new NIC();

  char test[4] = {'t', 'e', 's', 't'};
  nic->send(NIC::BROADCAST, 1, &test, 4);
}
```

- Compilar aplicação e gerar imagem do sistema
  - make APPLICATION=radio

Sistemas Operacionais para RSSF

- SO abstrai HW para aplicações
- Características do hardware de RSSF
  - Dimensões reduzidas: diminuir potência
  - Baixo consumo de energia:
    - Ligar apenas o que é usado
    - Desempenho reduzido geralmente aceitável
    - Modos de operação de baixo consumo
  - Modularidade
    - Hardware bastante heterogêneo
    - Necessidade de abstrair diferenças arquiteturais

Sistemas Operacionais para RSSF

- Características do hardware de RSSF (cont.)
  - Canal de comunicação adaptável
    - independente da configuração do canal
    - funcionalidades básicas devem ser mantidas
    - configuração não deve afetar semântica da aplicação, apenas qualidade

Sistemas Operacionais para RSSF

- Requisitos de SO para RSSF [Wanner and Fröhlich 2008]
  - Funcionalidade básica de SO
  - Gerenciamento do consumo de energia
  - Reprogramação em campo
  - Abstração uniforme de sensores heterogêneos
  - Pilha de protocolos de comunicação configurável
  - Operação sob restrição de recursos

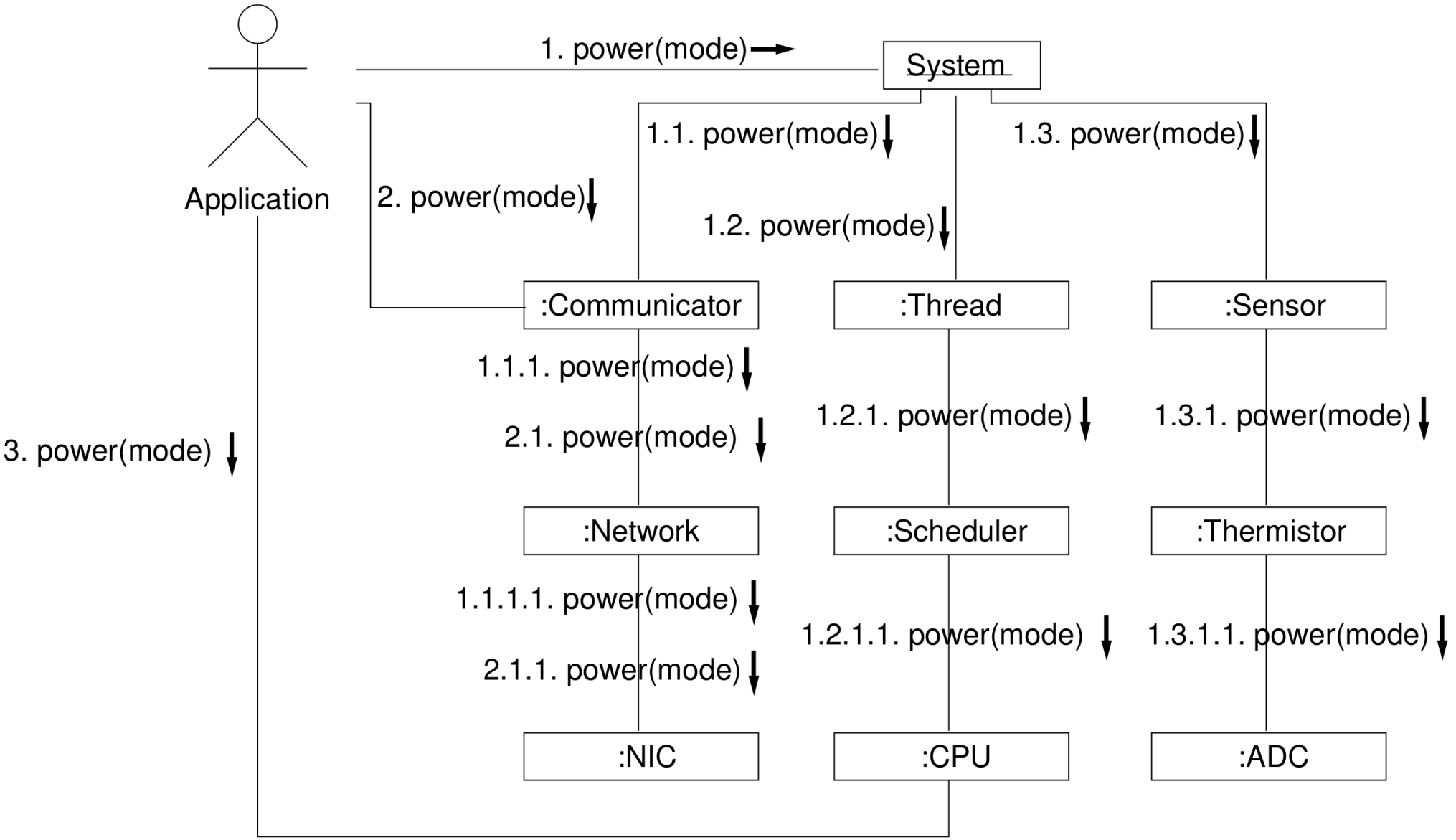
Funcionalidades básicas de SO

- Abstração de hardware
  - Interfaces padrão para aplicação (API)
  - Garantir portabilidade entre plataformas
- Gerenciamento de processos
  - Geralmente no modelo “monotarefa, multi-thread”
  - Escalonamento de “tempo real”
- Gerenciamento de memória
  - Alocação dinâmica com recursos restritos?
- Serviços de temporização
  - Para escalonamento
  - Tarefas de natureza periódica

Gerenciamento do consumo de energia

- Bateria
  - Vida útil da rede = vida útil da bateria
    - *“uma corrente é tão forte quanto seu elo mais fraco”*
  - Fator limitante no dimensionamento do sistema
- Sistema operacional deve
  - Manter ligados apenas dispositivos necessários
  - Complexidade deve ser abstraída da aplicação

Gerência de energia no EPOS



Reprogramação em campo

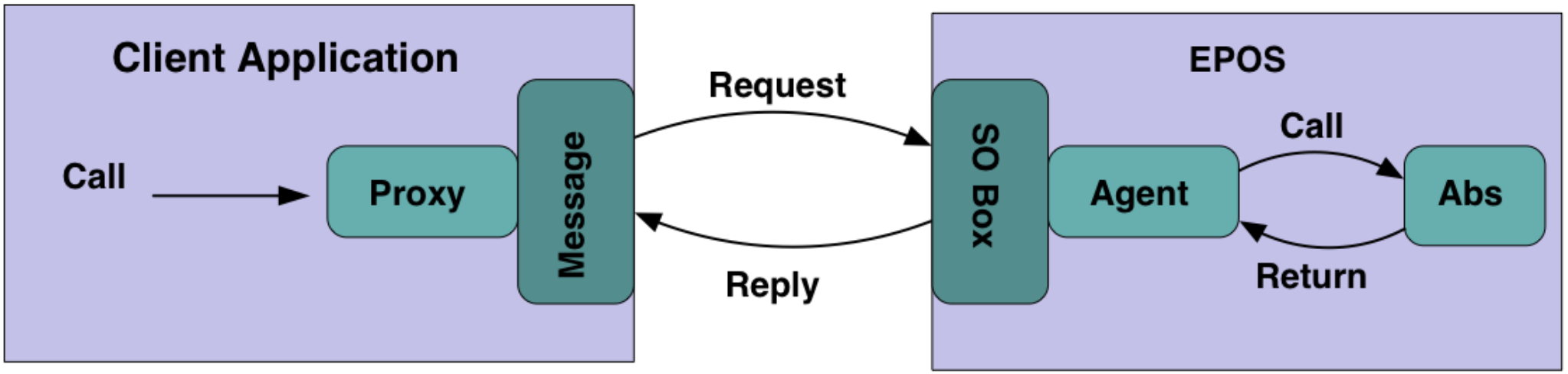
- RSSF
  - Grande volume de nodos
  - Em regiões inóspitas
  - Requisitos e/ou parâmetros de operação mudam
- A reprogramação de uma rede já implantada se torna essencial!
  - Prolonga vida útil do sistema
  - Permite a correção de bugs

Reprogramação em campo (cont.)

- A reprogramação deve ser utilizada com cautela
  - Gera um “stall” na rede, durante propagação
    - Rede inoperante durante reprogramação
  - Ineficiência energética
    - Uso excessivo do rádio
    - Escritas na memória de programa (EEPROM/Flash)

ELUS: EPOS Live Update System

- Reprogramação através da interface de comunicação
  - De todo o firmware
  - De componentes
  - De funções

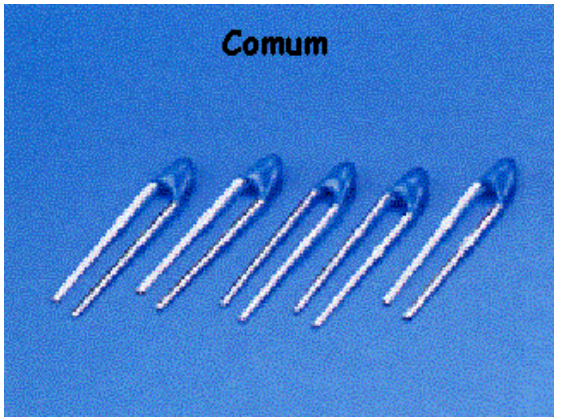


Abstração uniforme de sensores heterogêneos

- Modularidade => heterogeneidade
- Portabilidade comprometida
- CPU e rádio variam entre plataformas

- Mas sensores variam muito mais
  - Interface de acesso
  - Características operacionais
  - Curvas de resposta

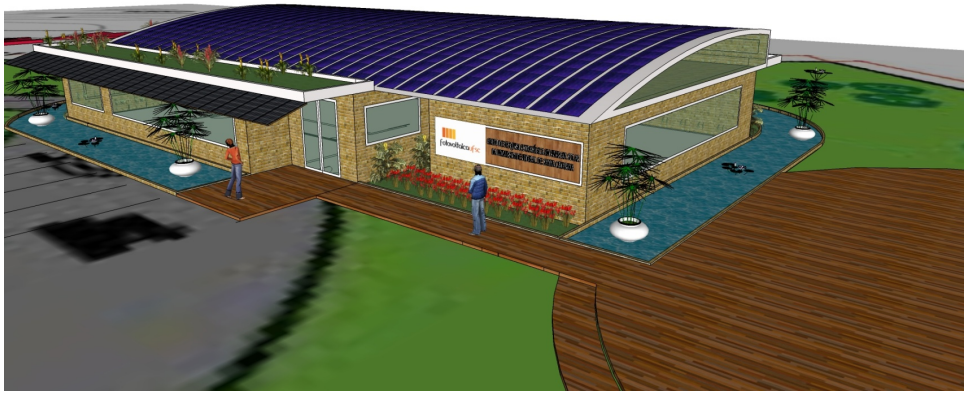
X

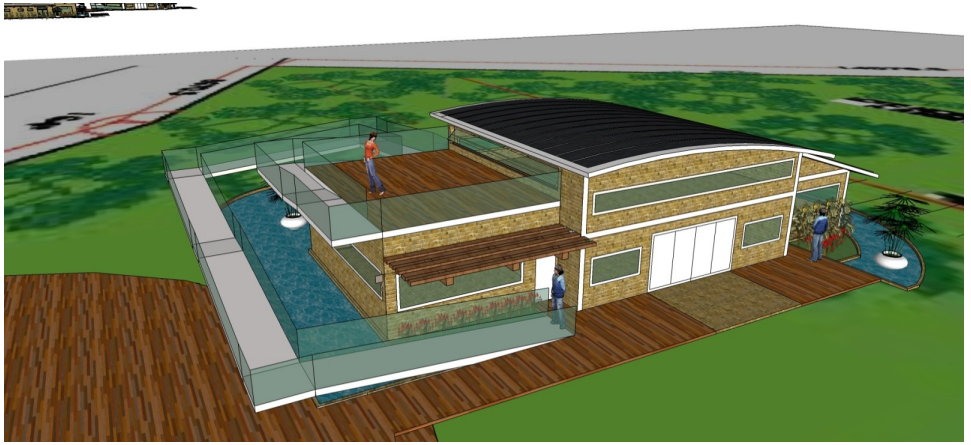


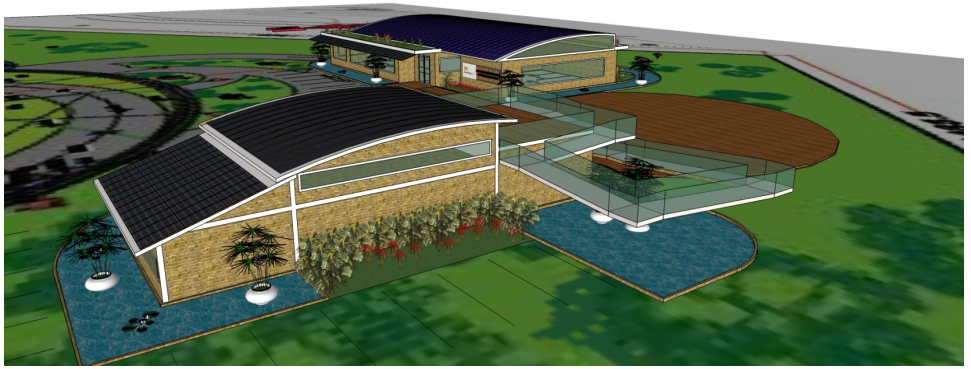
OpenEPOS e EPOSMote abertos

- É da comunidade!
  - http://epos.lisha.ufsc.br
- Suporte do LISHA
  - Interesse em melhorar tanto hardware quanto software
  - Novos projetos a caminho

Prédio Solar







CIA2

- Construindo Cidades Inteligentes: da Instrumentação de Ambientes ao desenvolvimento de Aplicações
- 18 universidades brasileiras

Vamos programar um pouco?

- Abram $EPOS/app/mc13224v_app.cc
- Vamos modificá-lo
  - Implementar novo protocolo

Mensagem	Layout
Temperatura	<NODE=0x0?> + <ID=0x00> + <16 bits>
Nível de bateria	<NODE=0x0?> + <ID=0x01> + <16 bits>

  - Utilizar 2 Periodic_Threads
    - 1 informando temperatura a cada 2s
    - 1 informando nível de bateria a cada 5s

Concluindo

- EPOS e EPOSMote
  - Plataforma aberta para RSSF
  - Sistema Operacional baseado em componentes
  - Hardware de sensoriamento IEEE 802.15.4 modular

- Vistem nosso site, usem e contribuam!

  - http://epos.lisha.ufsc.br

Obrigado!