An Introduction to Wireless Sensor Networks

CISTER Summer Internship 2017

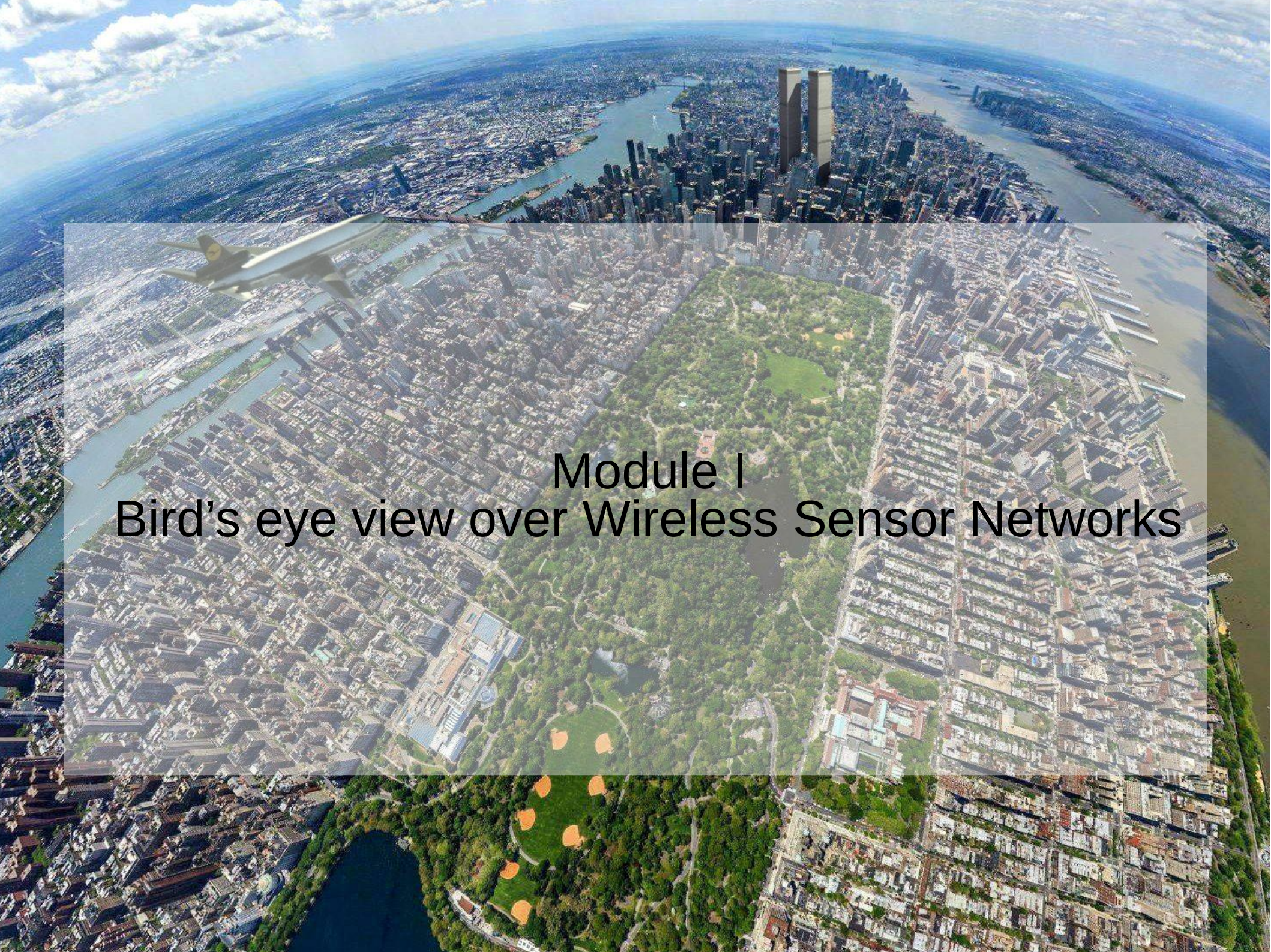
Ricardo Severino (PhD)
Research Associate

CISTER
Research Center in
Real-Time & Embedded
Computing Systems

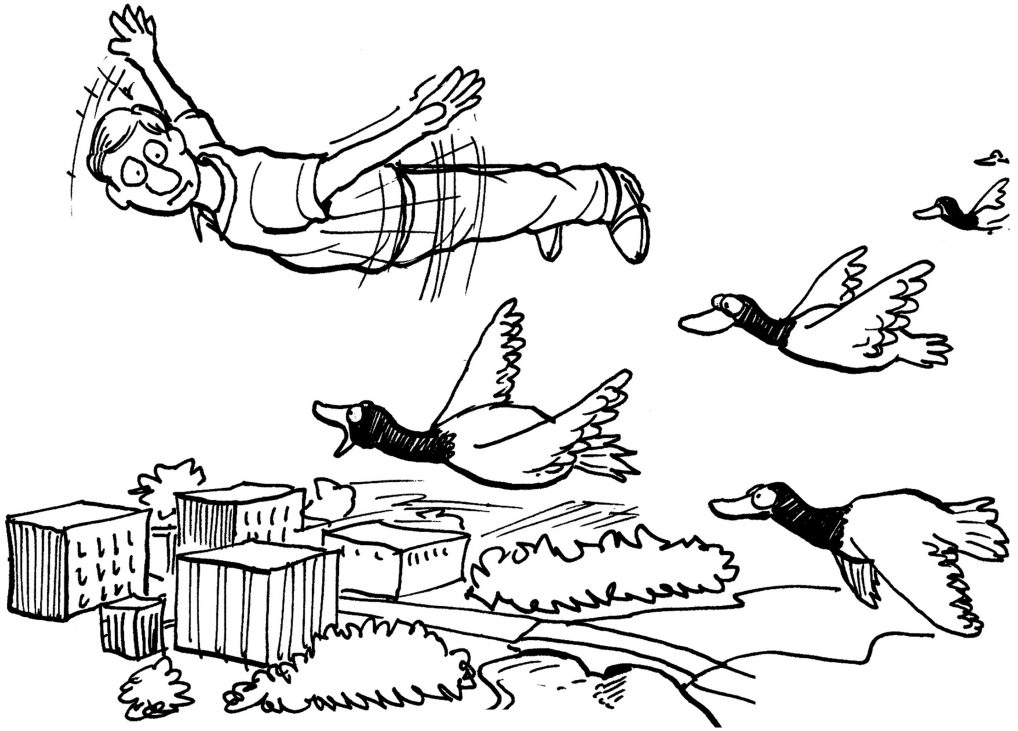
Outline

The lecture is split into 3 modules:

- Module I – Bird's eye of Wireless Sensor Networks
  - Overview of Wireless Sensor Networks
  - Motivation for WSN Applications
- Module II – Technologies and Tools
  - Anatomy of a WSN node (HW Concepts)
  - WSN Operating Systems
  - Communication Architectures
- Module III – Application Examples (Real Deployments@CISTER.ISEP)
  - Extra WSN Concepts
  - Structural Health Monitoring
  - Datacenter Monitoring
  - ArtWise Robot Testbed
  - WSN@CISTER



Module I
Bird's eye view over Wireless Sensor Networks



"Sometimes it's good to get a different perspective."

Defining WS(A)Ns

- what are “wireless sensor/actuator networks”?
  - “wireless (communication)”
    - “wireless communication” is the transfer of information over a distance without the use of electrical conductors or "wires“ using some form of energy, e.g. radio frequency (RF), infrared light (IR), laser light, visible light, acoustic energy
  - “sensor”
    - a sensor is a device that measures a physical quantity and converts it into a signal which can be read by an observer or by an instrument, e.g. thermocouple, strain gauge

Defining WS(A)Ns

- what are “wireless sensor/actuator networks”? (cont.)
  - “actuator” devices which transform an input signal (mainly an electrical signal) into motion
    - e.g. electrical motors, pneumatic actuators, hydraulic pistons, relays, electro-valves, piezoelectric actuators, buzzers, lamps
  - “network”
    - a “computer network” is a group of interconnected computers

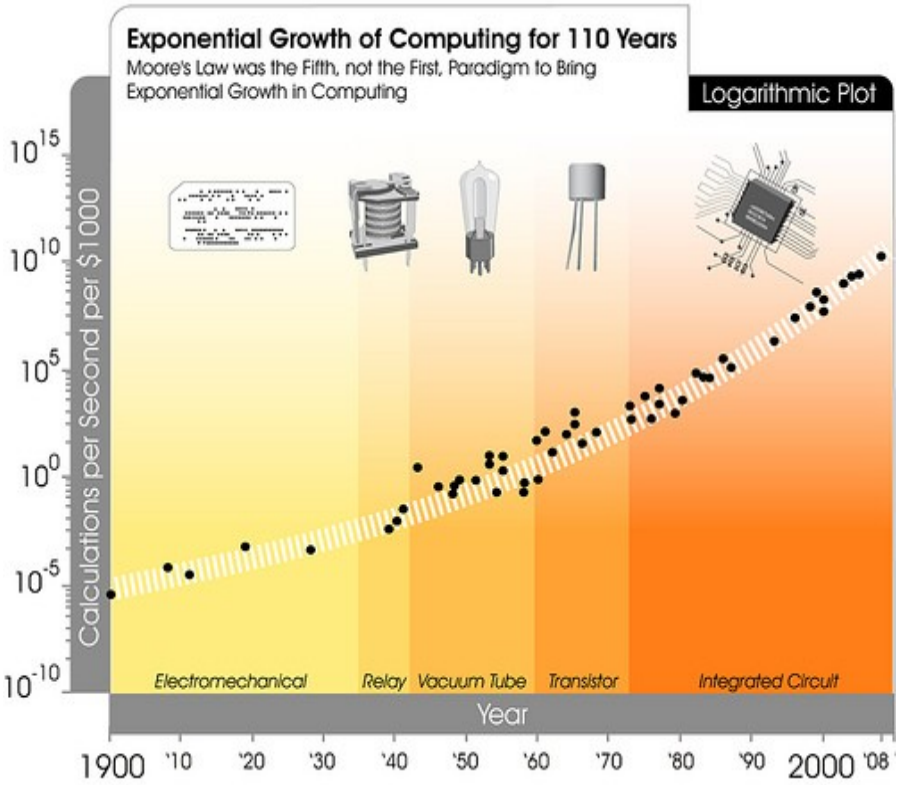
Defining WS(A)Ns

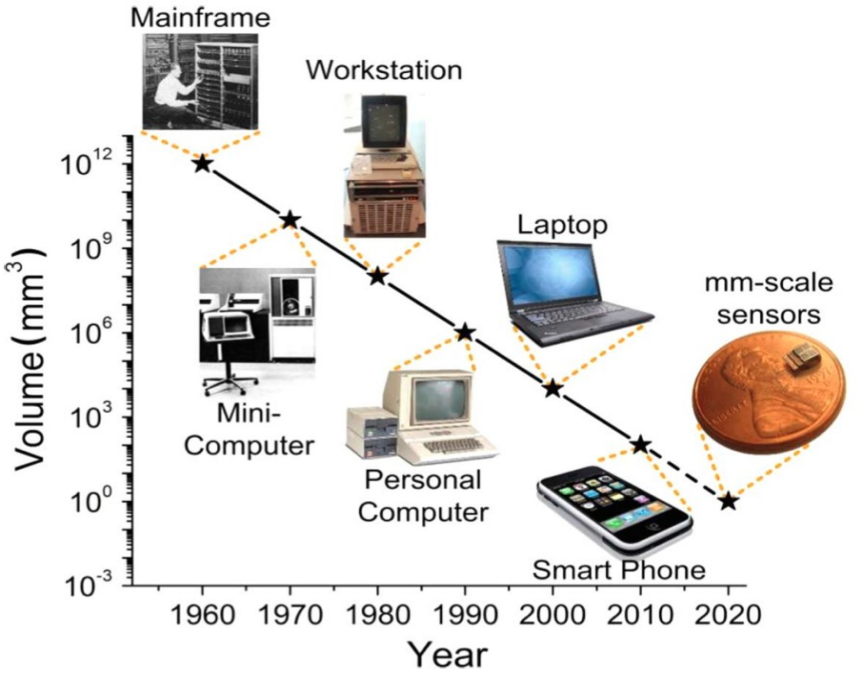
- So… what are “wireless sensor/actuator networks”?
  - a wireless sensor network (WSN) is a wireless network consisting of spatially distributed autonomous devices using sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants, at different locations.
  - originally motivated by military applications such as battlefield surveillance; now used in many civilian application areas, including environment and habitat monitoring, healthcare applications, home automation, and traffic control

How did we get there? Microelectronics Revolution

The Semi-conductor => Micro-electronics => Micro-controller => Embedded Systems

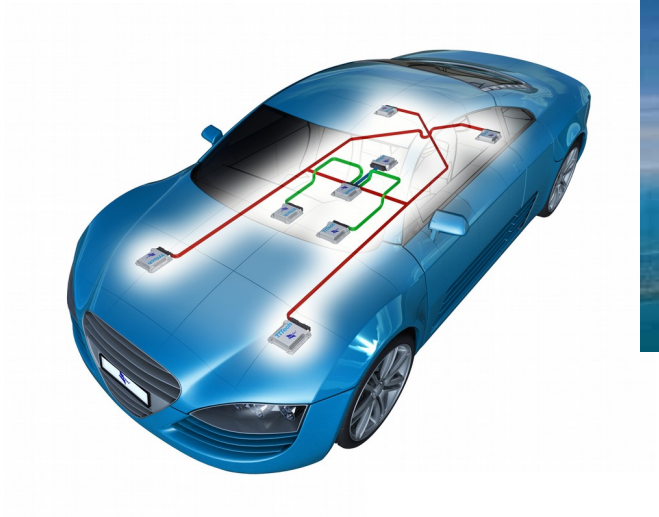
Moore's law (1965) is the observation that the number of transistors in a dense integrated circuit doubles approximately every two years.





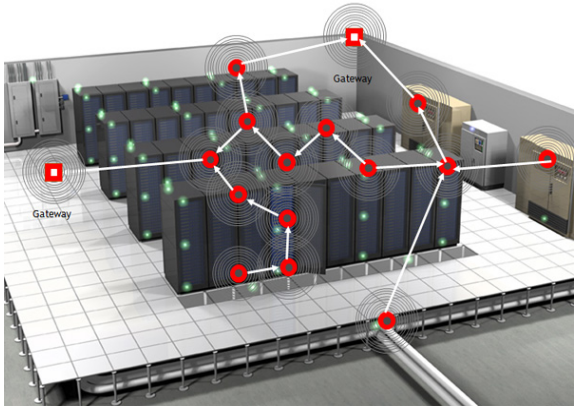
Faster, smaller, in everything, everywhere

The Semi-conductor => Micro-electronics => Micro-controller => Embedded Systems => Smart "everything"



Lets put sensors in it

- Pervasiveness and ubiquity of embedded systems + wireless technologies
- Eagerness to control and monitor everything, everywhere
- Wireless Sensor Networks
  - The Internet of Things (IoT)
  - Cyber-Physical Systems (Timeliness in environmental monitoring/industrial automation and process control)
    - Tighter interaction between sensing and actuation
    - Communications must be logically correct but also produced on time

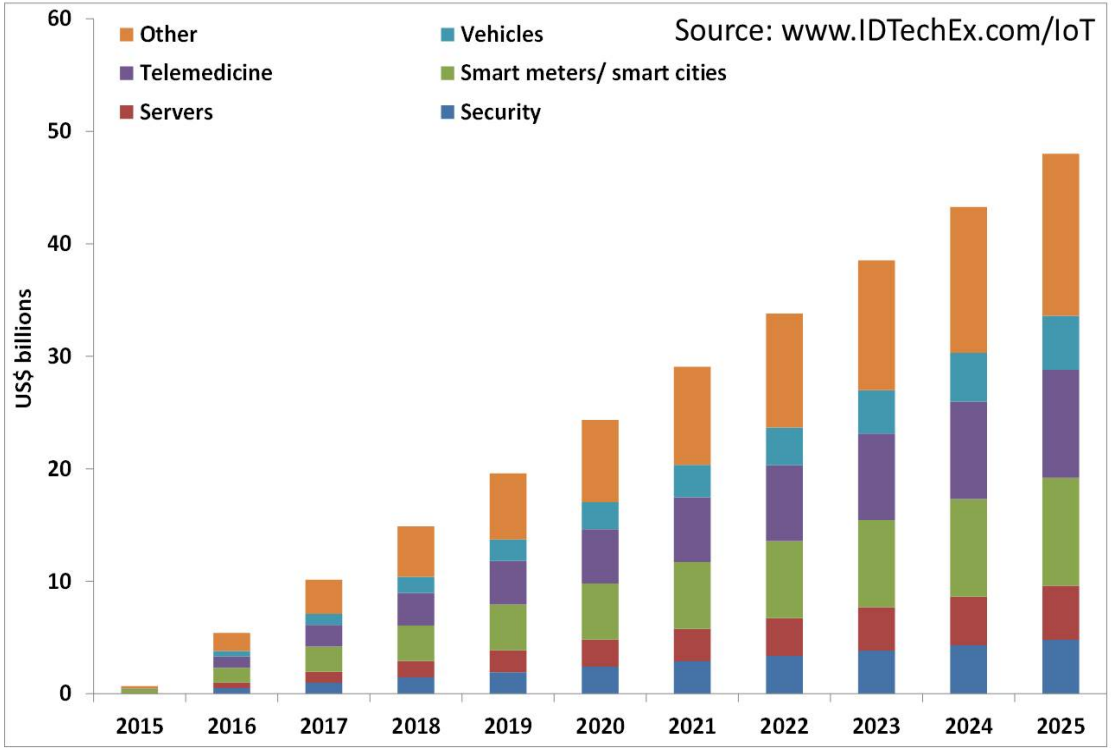


History of Wireless Sensor Networks

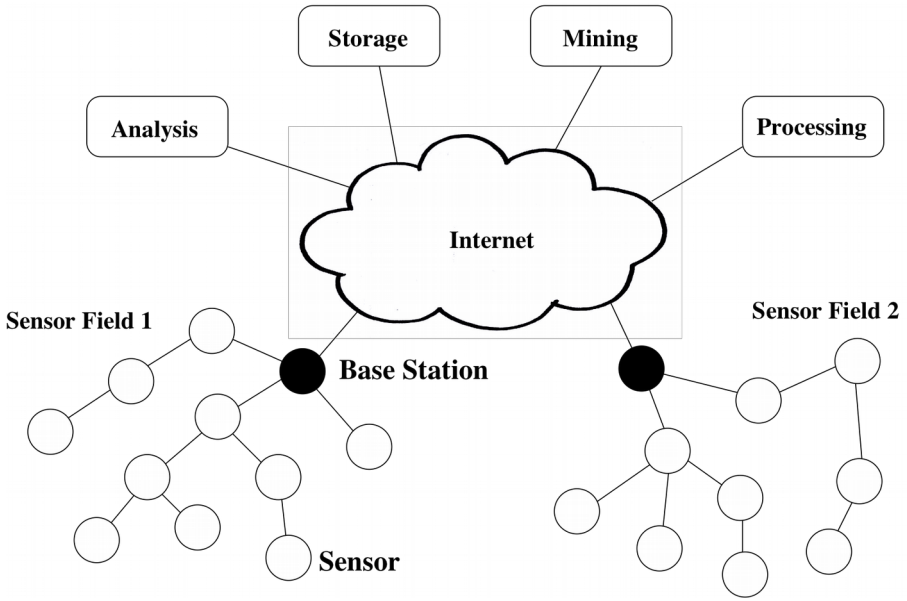
- DARPA: Distributed Sensor Nets Workshop (1978)
  - Distributed Sensor Networks (DSN) program (early 1980s)
  - Sensor Information Technology (SensIT) program
- UCLA and Rockwell Science Center
  - Wireless Integrated Network Sensors (WINS)
  - Low Power Wireless Integrated Microsensor (LWIM) (1996)
- UC-Berkeley
  - Smart Dust project (1999)
  - concept of "motes": extremely small sensor nodes
- Berkeley Wireless Research Center (BWRC)
  - PicoRadio project (2000)
- MIT
  - μAMPS (micro-Adaptive Multidomain Power-aware Sensors) (2005)

Market forecasts...

- Cisco IBSG predicts 25 billion IoT devices by 2015 and 50 by 2020;
- IDTechEx market value for IoT IP-addressed sensing nodes to grow from less than $1 Billion (US) in 2015 to greater than $48 Billion (US) by 2025.



General System Model of a WSN



- Multiple sensors (often hundreds or thousands) form a network to cooperatively monitor large or complex physical environments
- Acquired information is wirelessly communicated to a base station (BS), which propagates the information to remote devices for storage, analysis, and processing

WSN types

- from “traditional” WSN to other “forms”:
  - Body Sensor Networks (BSN)
  - Vehicular Sensor Networks (V2V, V2I)
  - Machine-to-Machine (M2M)
  - Underwater (Acoustic) Sensor Networks (UW-ASN)
  - Internet-of-Things (pervasive Internet)
  - Urban/social/participatory Sensor Networks
  - Interplanetary Sensor Networks
  - Industrial Internet/Industry 4.0
  - Near-Field Communications (NFC)

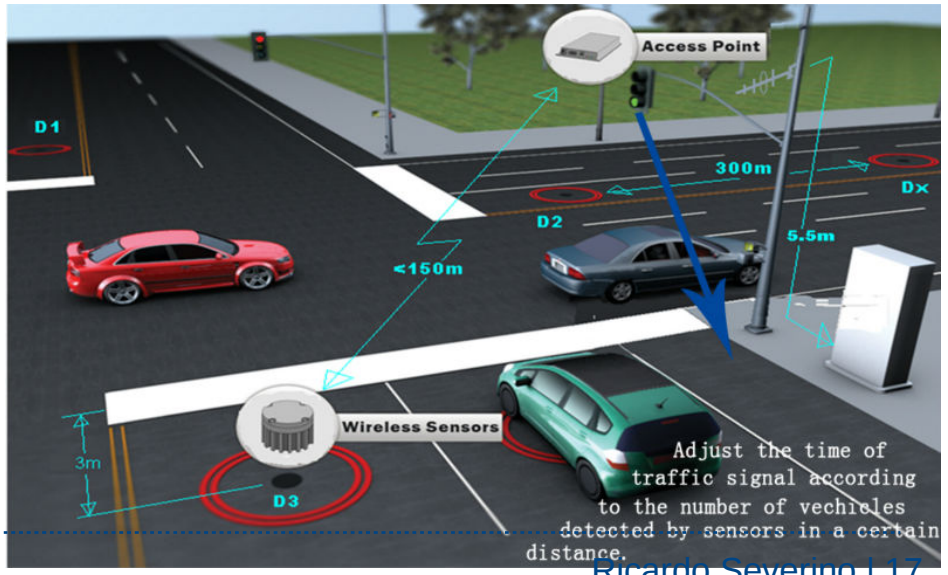
Motivation for WSN Application Areas

Traffic Management

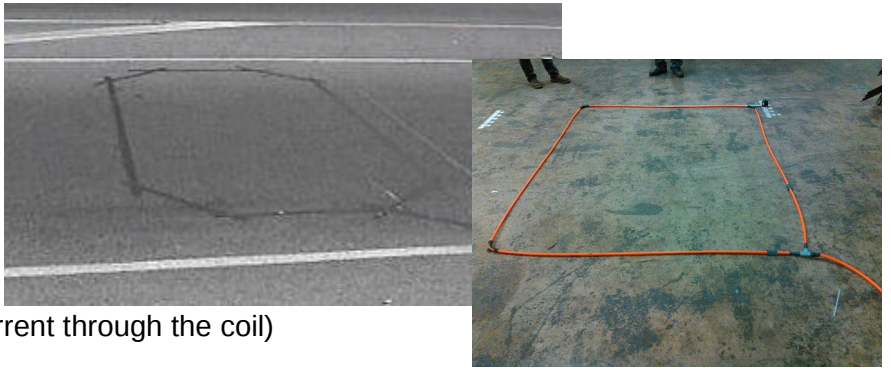
- Motivation:
  - ground transportation is a vital and a *complex* socio-economic infrastructure
  - it is *linked* with and provides *support* for *a variety of systems*, such as supply-chain, emergency response, and public health
  - the 2009 Urban Mobility Report reveals that in 2007, congestion caused urban Americans to
    - travel *4.2 billion hours more*
    - purchase *an extra 2.8 billion gallons of fuel*
  - congestion *cost is very high* - $87.2 billion; an increase of more than 50% over the previous decade

Traffic Management

- Motivation:
  - building new roads is *not* a feasible solution for many cities
    - lack of free space
    - high cost of demolition of old roads
  - *one approach*: put in place distributed systems that reduce congestions
    - *gather information* about the density, sizes, and speed of vehicles on roads
    - *infer congestions*
    - *suggest alternative routes* and emergency exits

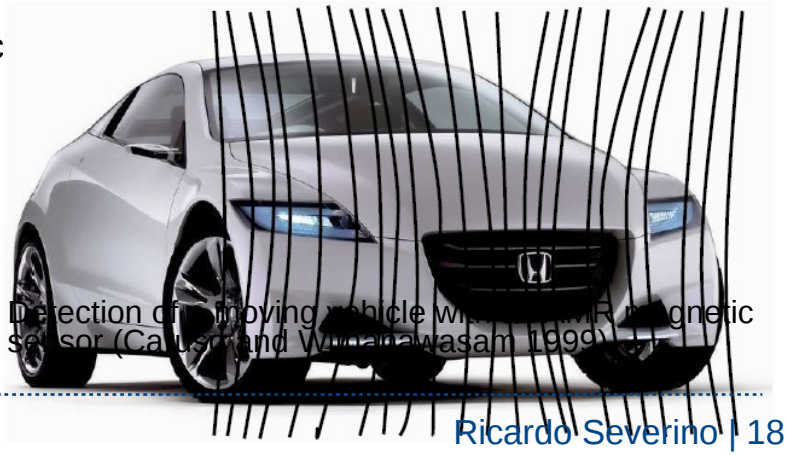


Traffic Management



- How to sense?
  - Inductive loops (in-road sensing devices)
    - *Advantages:*
      - unaffected by weather
      - provide direct information (few ambiguity)
    - how does it work: using *Faraday's induction law*
      - a coil of wire (several meters in diameter, passes an electric current through the coil)
      - buried under the road and connected to a roadside control box
      - magnetic field strength can be induced as a result of a current and the speed and the size of passing vehicles
  - Magnetic sensors can determine the *direction* and *speed* of a vehicle
    - a moving vehicle can disturb the distribution of the magnetic field
      - by producing its own magnetic field
      - by cutting across it
    - The magnitude and direction of the disturbance depends on
      - the *speed*, *size*, *density* and *permeability* of the vehicle
    - Classification of magnetic sensors:
      - *low field* (below 1µGauss)
      - *medium field* (between 1µGauss and 10µGauss)
      - *high field* (above 10µGauss)
    - The *concentration* of magnetic flux *varies as* the *vehicle moves*; it c
    - The field variation reveals a *detailed* magnetic signature
    - It is possible to distinguish between different types of vehicles

PM-04 magnetic field sensors from Alvi Technologies



Detection of a moving vehicle with [illegible] magnetic sensor (Caruso and Withanawasam 1999)

Traffic Management

- Arora et al. (2004) - A Line in the Sand
  - Deploys *90 sensor nodes* to detect the movement of vehicles and people (e.g., soldiers)
  - *78* of the nodes were *magnetic sensor nodes* that were deployed in a 60×25 square foot area
  - *12 radar sensor nodes* were overlaid on the network
  - These nodes form a *self-organizing* network which connects itself to a remote computer via a base station and a long haul radio repeater

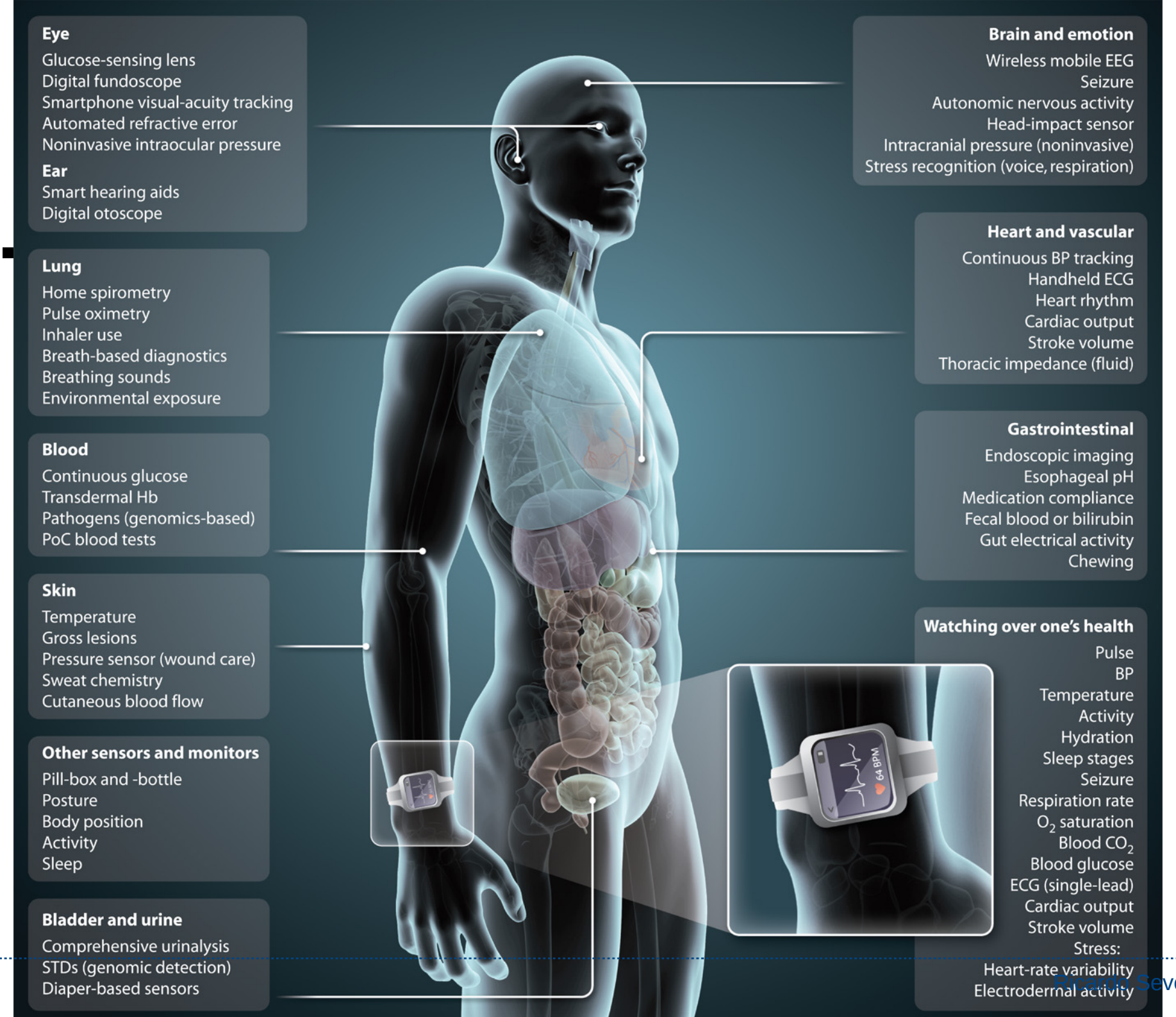
Fig. 6. Sensor hardware from left to right: (a) Mica2 network node, (b) Mica Sensor Board, (c) Mica Power Board, (d) TWR-ISM-002 Radar Board, and (e) All of the boards attached together.

Health Care

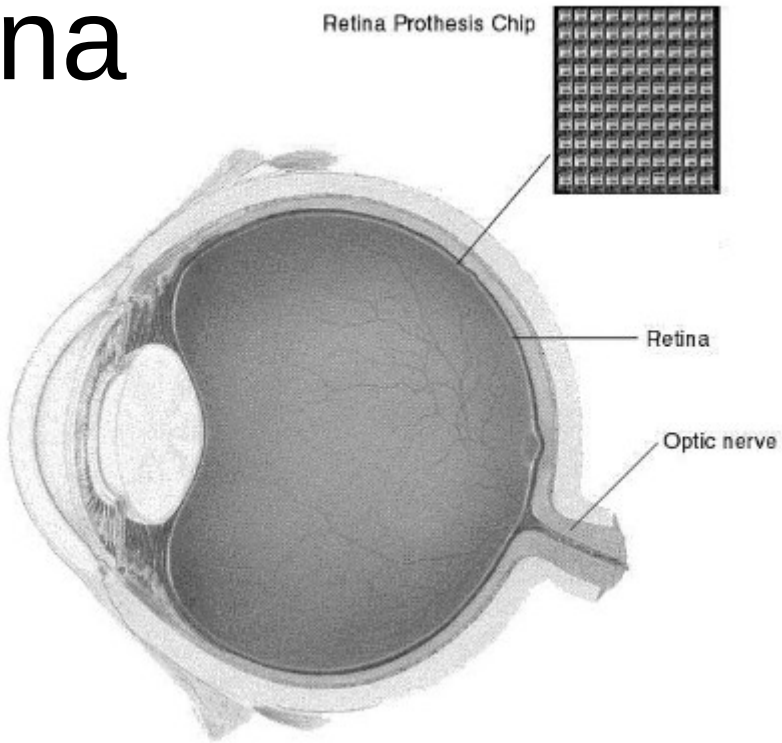
- A wide range of health care applications have been proposed for WSN, including *monitoring patients* with:
  - Parkinson's Disease and epilepsy
  - heart patients
  - patients rehabilitating from stroke or heart attack
  - elderly people
- Health care applications do *not* function as *standalone* systems
- They are *integral parts* of a comprehensive and complex health and rescue system
- Preventive health care - to reduce health spending and mortality rate
  - but some patients find certain practices *inconvenient, complicated,* and *interfering* with their daily life (Morris 2007)
  - many *miss* checkup visits or therapy sessions because of *a clash of schedules with* established living and working *habits*, *fear* of overexertion, or transportation *cost*

Health Care

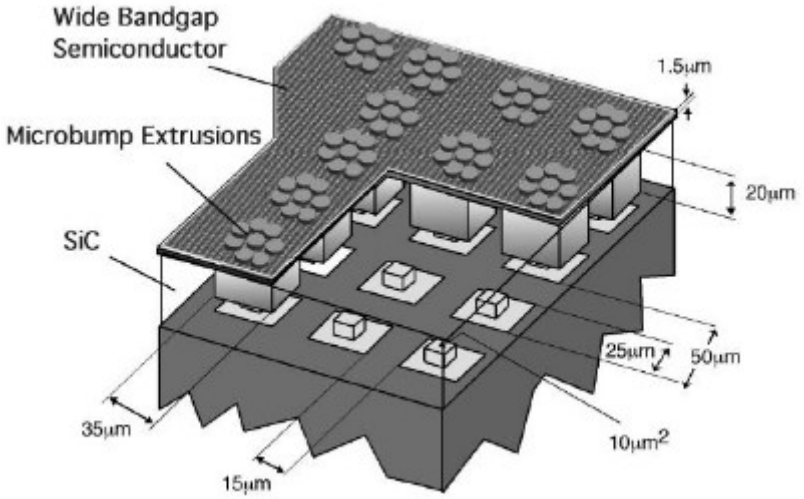
- To deal with these problems, researchers proposed comprehensible solutions that involve the following tasks:
  - building *pervasive systems* that *provide* patients with *rich information* about diseases and their prevention mechanisms
  - seamless integration of health infrastructures with *emergency and rescue operations* as well as *transportation systems*
  - developing reliable and unobtrusive health *monitoring systems* that can be worn by patients to *reduce the task* and *presence of medical personnel*
  - *alarming* nurses and doctors *when* medical intervention is *necessary*
  - *reducing inconvenient* and *costly check-up visits* by *creating reliable links* between autonomous health monitoring systems and health institutions deal with these problems, researchers proposed comprehensible solutions that involve the following tasks:


Eye
Glucose-sensing lens
Digital fundoscope
Smartphone visual-acuity tracking
Automated refractive error
Noninvasive intraocular pressure
Ear
Smart hearing aids
Digital otoscope
Lung
Home spirometry
Pulse oximetry
Inhaler use
Breath-based diagnostics
Breathing sounds
Environmental exposure
Blood
Continuous glucose
Transdermal Hb
Pathogens (genomics-based)
PoC blood tests
Skin
Temperature
Gross lesions
Pressure sensor (wound care)
Sweat chemistry
Cutaneous blood flow
Other sensors and monitors
Pill-box and -bottle
Posture
Body position
Activity
Sleep
Bladder and urine
Comprehensive urinalysis
STDs (genomic detection)
Diaper-based sensors
Brain and emotion
Wireless mobile EEG
Seizure
Autonomic nervous activity
Head-impact sensor
Intracranial pressure (noninvasive)
Stress recognition (voice, respiration)
Heart and vascular
Continuous BP tracking
Handheld ECG
Heart rhythm
Cardiac output
Stroke volume
Thoracic impedance (fluid)
Gastrointestinal
Endoscopic imaging
Esophageal pH
Medication compliance
Fecal blood or bilirubin
Gut electrical activity
Chewing
Watching over one's health
Pulse
BP
Temperature
Activity
Hydration
Sleep stages
Seizure
Respiration rate
O_2 saturation
Blood CO_2
Blood glucose
ECG (single-lead)
Cardiac output
Stroke volume
Stress:
Heart-rate variability
Electrodermal activity
64 BPM

Artificial Retina



- Schwiebert et al. (2001) developed a micro-sensor array that can be *implanted in the eye* as an artificial retina to assist people with visual impairments
- The system consists of *an integrated circuit* and *an array of sensors*
- An integrated circuit
  - is coated with a *biologically inert substan*
  - is *a multiplexer* with on-chip switches and pads to support a 10×10 grid of connections; it operates at 40KHz
  - has an *embedded transceiver* for wired a wireless communications
  - each *connection* in the chip interfaces a sensor through an aluminum probe surfa

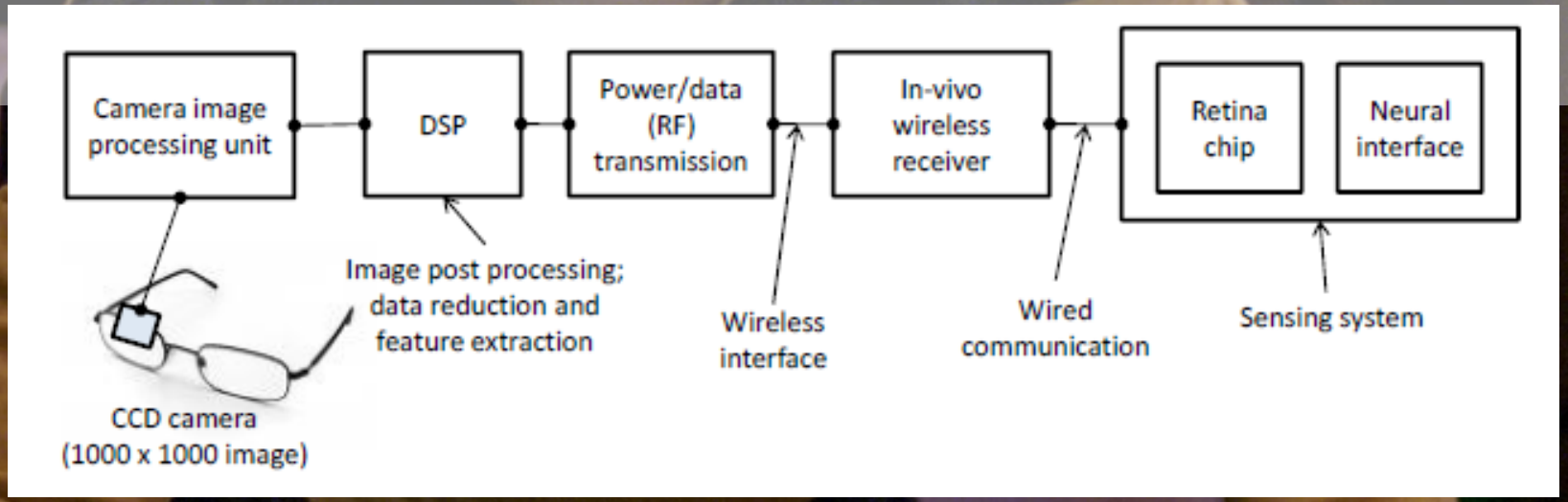


Artificial Retina

- An array of sensors
  - each sensor is *a micro-bump*, sufficiently *small and light*
  - the *distance* between adjacent micro-bumps is approximately *70 microns*
  - the sensors *produce electrical signals* proportional to the light reflected from an object being perceived
  - the *ganglia* and additional tissues *transform the electrical energy* into *a chemical energy*
  - the chemical energy is *transformed* into *optical signals* and *communicated to the brain* through the optical nerves
  - the *magnitude and wave shape* of the transformed energy *corresponds to* the response of *a normal retina to light stimulation*

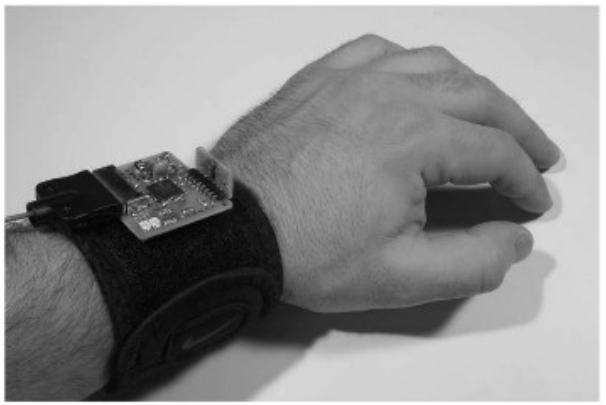
Artificial Retina

- The signal processing steps of the artificial retina
  - a *camera* embedded in a pair of spectacles *directs its output to* a real-time *DSP*
  - the camera can be combined with a laser pointer for automatic focusing
  - *the output* of the DSP *is compressed* and *transmitted* through a wireless link to the *implanted sensor array*
  - the sensor array decodes the image and produces a corresponding electrical signal



Parkinson's Disease

- The aim is to augment or entirely replace a human observer and to help physicians fine-tune medication dosage
- **Weaver** (2003)
  - the system consists of
    - a lightweight sensor node with 3D accelerometer sensors (sampled at a rate of 40Hz.)
    - a processor core
    - a storage system for logging data for latter retrieval
  - the system could record 17 hours of accelerometer data
  - the patients wear the nodes in their ankles and wrists
  - the report reveals that the system was able to identify the occurrence of dyskinesia at the rate of 80%

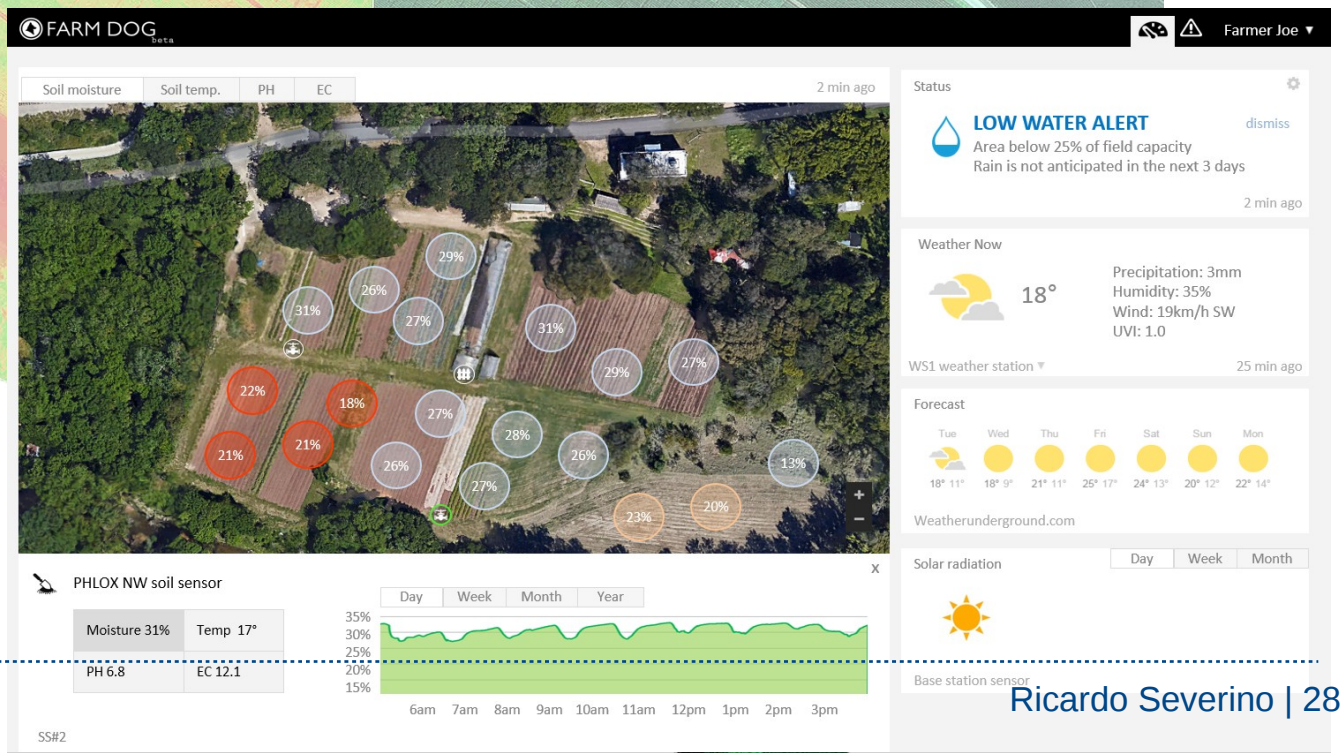


Precision Agriculture

- Motivation:
  - traditionally, a large farm is taken as homogeneous field in terms of *resource distribution* and its response to *climate change*, *weeds*, and *pests*
  - accordingly, farmers administer
    - *fertilizers*, *pesticides*, *herbicides*, and *water resources*
  - in reality, wide spatial diversity in *soil types*, *nutrient content*, and other important factors
  - therefore, treating it as a uniform field can cause
    - *inefficient use of resources*
    - *loss of productivity*
- Precision agriculture is a method of farm management that enables farmers to produce *more efficiently* through *a frugal use of resources*

Precision Agriculture

- Precision agriculture *technologies*:
  - yield monitors
  - yield mapping
  - variable rate fertilizer
  - weed mapping
  - variable spraying
  - topography and boundaries
  - salinity mapping
  - guidance systems



Wine Vinyard (2004)

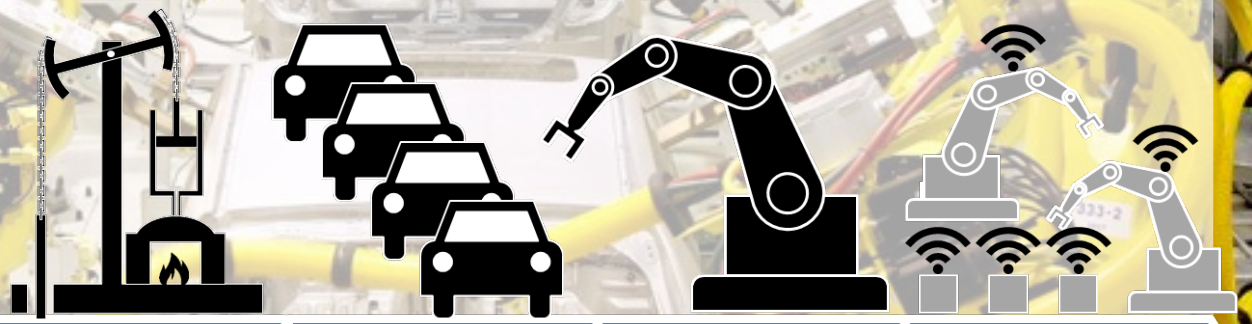
- Beckwith et al. deploy a WSN to *monitor* and characterize variation in *temperature* of a wine vineyard
  - heat summation and periods of freezing temperatures
- *65 nodes* in a grid like pattern 10 to 20 meters apart, covering about two acres
- *Easy to develop* the network (1 person day)
  - due to the self-configuration nature of the network
  - inherent structured layout of vineyard fields
- Two essential constraints of the network topology
  - placement of nodes in an area of viticulture interest
  - the support for multi-hop communication

Wine Vinyard (2004)

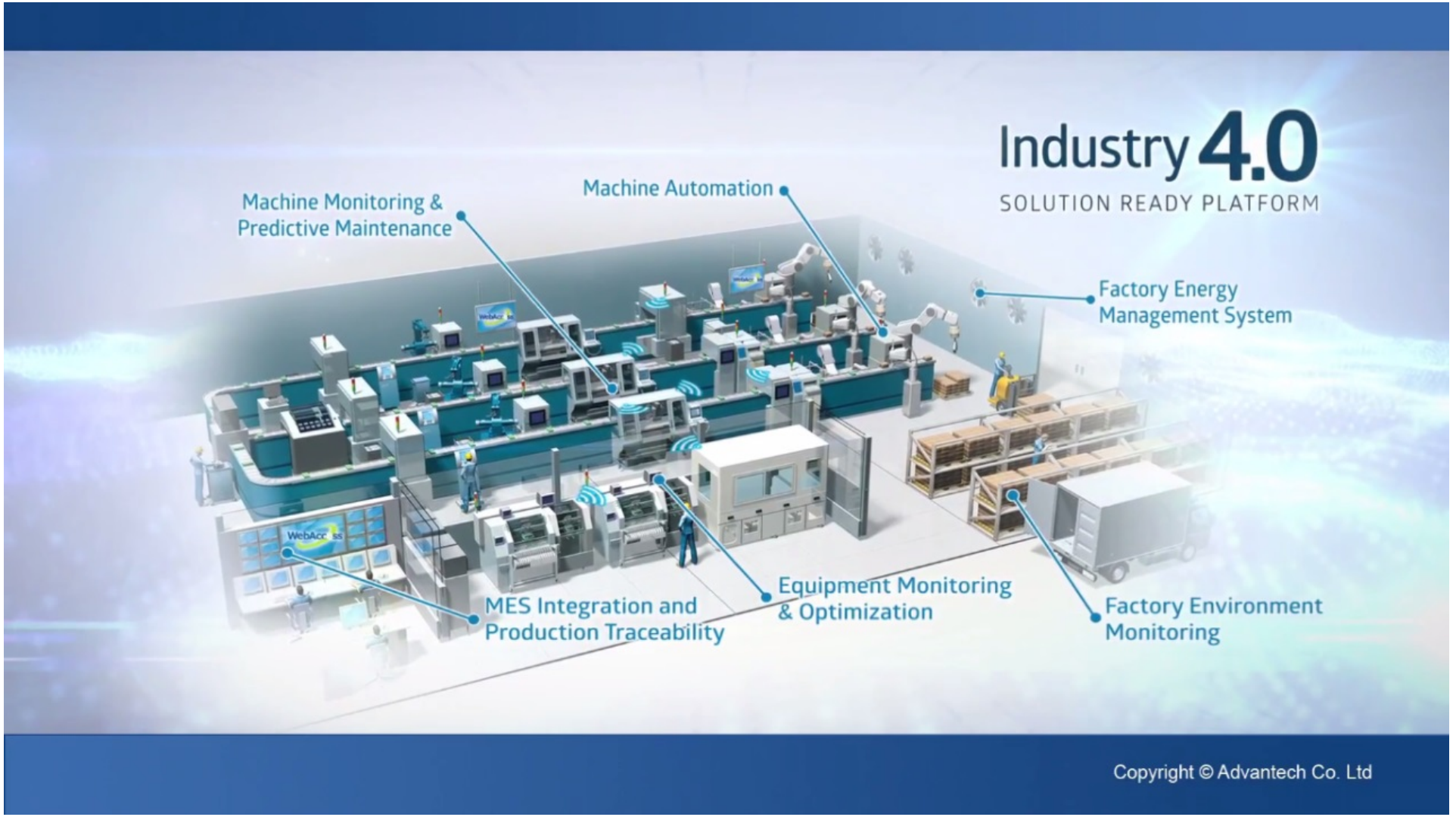
- The data were used to investigate several aspects:
  - the existence of *co-variance between the temperature* data collected by the network
  - *growing degree day differences*
  - *potential frost damage*
- The *mean data* enabled to observe the relative differences between heat units accumulation during that period
  - according to the authors' report, the extent of variation in this vineyard – there was a measured difference of over 35% of heat summation units (HSUs) in as little as 100 meters

Industrial Automation/Process Control

- Trends in automation and manufactoring technology towards high flexibility and strong costumization of products.
  - Industry 4.0 (Europe)
  - Industrial Internet
- Other goals
  - condition monitoring and fault diagnosis -components and systems are able to gain self-awareness and self-predictiveness.
  - Factory floor monitoring
- Technologies
  - Cyber Physical Systems
  - Internet of Things
  - Cloud Computing
  - Autonomous Robots
  - Augmented reality



1st	2nd	3rd	4th
Mechanization, water power, steam power	Mass production, assembly line, electricity	Computer and automation	Cyber Physical Systems


Industry 4.0
SOLUTION READY PLATFORM
Machine Monitoring & Predictive Maintenance
Machine Automation
Factory Energy Management System
MES Integration and Production Traceability
Equipment Monitoring & Optimization
Factory Environment Monitoring
Copyright © Advantech Co. Ltd

Just a quick comment

- Different applications require different
  - network topologies (scalability)
  - set of sensors (heterogeneity)
  - time constraints (timeliness)
  - power sources (energy-efficiency)
  - security concerns (security)
- All of these are associated with a non-functional requirement.
  - requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors, e.g. reliability, robustness.
- We will talk about this later...

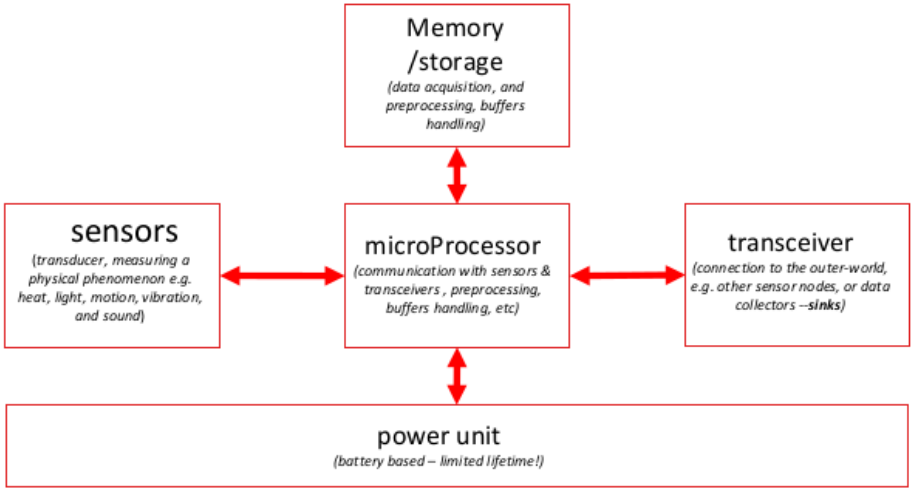
Also... this makes it challenging

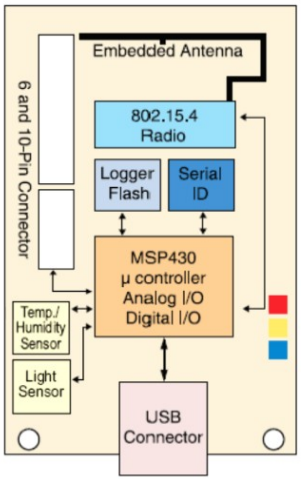
Traditional Networks	Wireless Sensor Networks
General-purpose design; serving many applications	Single-purpose design; serving one specific application (being extended with IoT :D)
Typical primary design concerns are network performance and latencies; energy is not a primary concern	Energy is the main constraint in the design of all node and network components
Networks are designed and engineered according to plans	Deployment, network structure, and resource use are often ad-hoc (without planning)
Devices and networks operate in controlled and mild environments	Sensor networks often operate in environments with harsh conditions
Maintenance and repair are common and networks are typically easy to access	Physical access to sensor nodes is often difficult or even impossible
Component failure is addressed through maintenance and repair	Component failure is expected and addressed in the design of the network
Obtaining global network knowledge is typically feasible and centralized management is possible	Most decisions are made localized without the support of a central manager

Module II
Technologies and Tools

Anatomy of a WSN Node

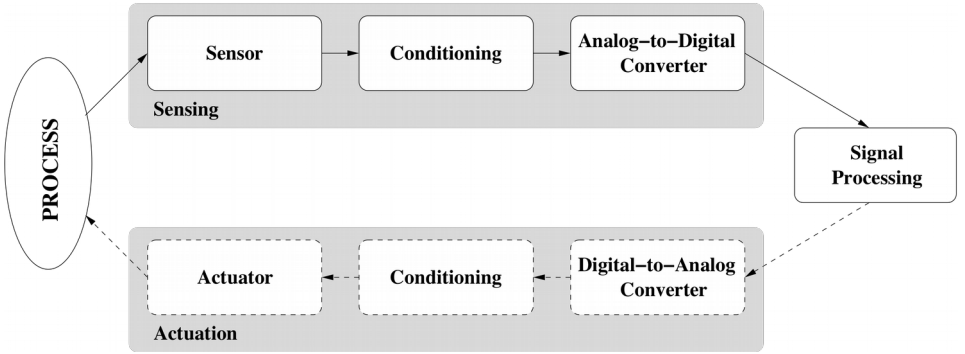
- Wireless sensor nodes are the essential building blocks in a wireless sensor network
  - *sensing*, *processing*, and *communication*
  - *stores* and *executes* the communication protocols as well as data processing algorithms
- The node consists of *sensing*, *processing*, *communication*, and *power subsystems*
  - trade-off between flexibility and efficiency – both in terms of energy and performance

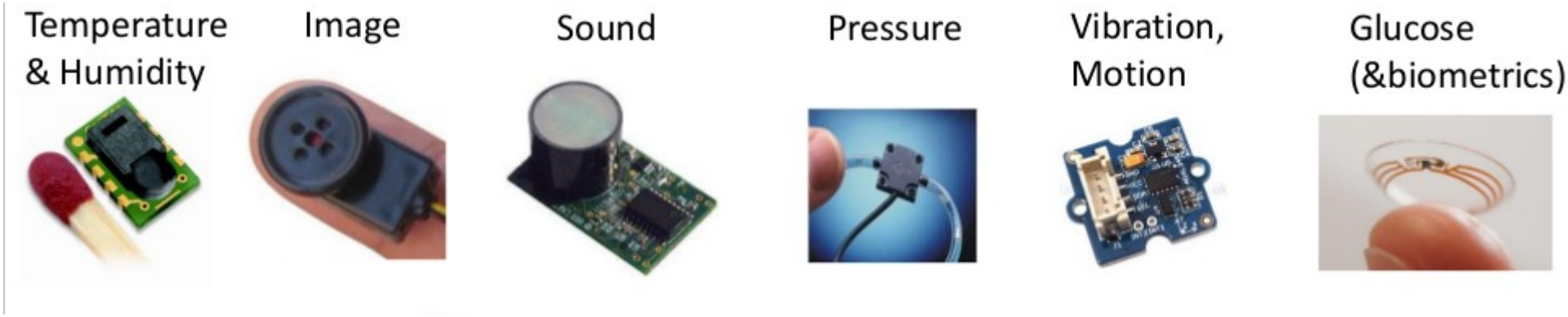




Sensing/actuation sub-system

- Sensors capture phenomena in the physical world (process, system, plant)
- Signal conditioning prepare captured signals for further use (amplification, attenuation, filtering of unwanted frequencies, etc.)
- Analog-to-digital conversion (ADC) translates analog signal into digital signal
- Digital signal is processed and output is often given (via digital-analog converter and signal conditioner) to an actuator (device able to control the physical world)





Classifying Sensors

- Physical property to be monitored determines type of required sensor

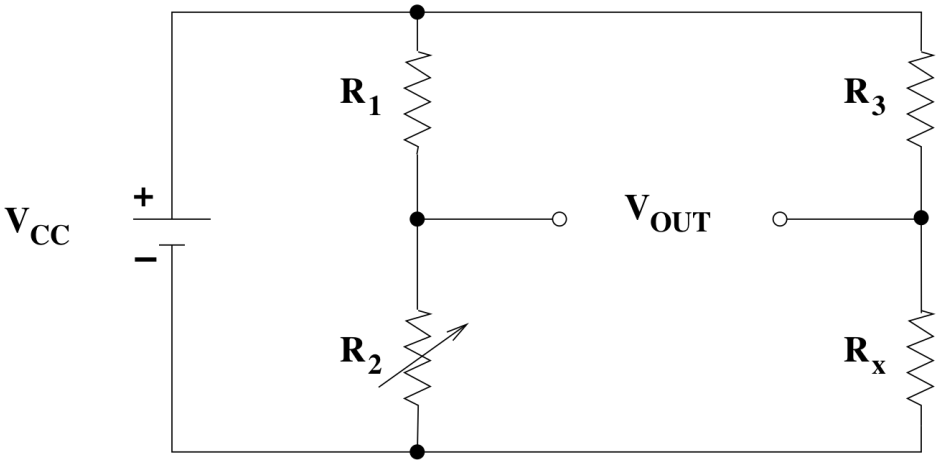
Type	Examples
Temperature	Thermistors, thermocouples
Pressure	Pressure gauges, barometers, ionization gauges
Optical	Photodiodes, phototransistors, infrared sensors, CCD sensors
Acoustic	Piezoelectric resonators, microphones
Mechanical	Strain gauges, tactile sensors, capacitive diaphragms, piezoresistive cells
Motion, vibration	Accelerometers, mass air flow sensors
Position	GPS, ultrasound-based sensors, infrared-based sensors, inclinometers
Electromagnetic	Hall-effect sensors, magnetometers
Chemical	pH sensors, electrochemical sensors, infrared gas sensors
Humidity	Capacitive and resistive sensors, hygrometers, MEMS-based humidity sensors
Radiation	Ionization detectors, Geiger-Mueller counters

Classifying Sensors

- Power supply:
  - active sensors require external power, i.e., they emit energy (microwaves, light, sound) to trigger response or detect change in energy of transmitted signal (e.g., electromagnetic proximity sensor)
  - passive sensors detect energy in the environment and derive their power from this energy input (e.g., passive infrared sensor)
- Electrical phenomenon:
  - resistive sensors use changes in electrical resistivity (ρ) based on physical properties such as temperature (resistance $R = \rho * l/A$)
  - capacitive sensors use changes in capacitor dimensions or permittivity (ε) based on physical properties (capacitance $C = \varepsilon * A/d$)
  - inductive sensors rely on the principle of inductance (electromagnetic force is induced by fluctuating current)
  - piezoelectric sensors rely on materials (crystals, ceramics) that generate a displacement of charges in response to mechanical deformation

Signal Conditioning Example Wheatstone Bridge Circuit

R1, R2, and R3 known (R2 adjustable)
Rx is unknown



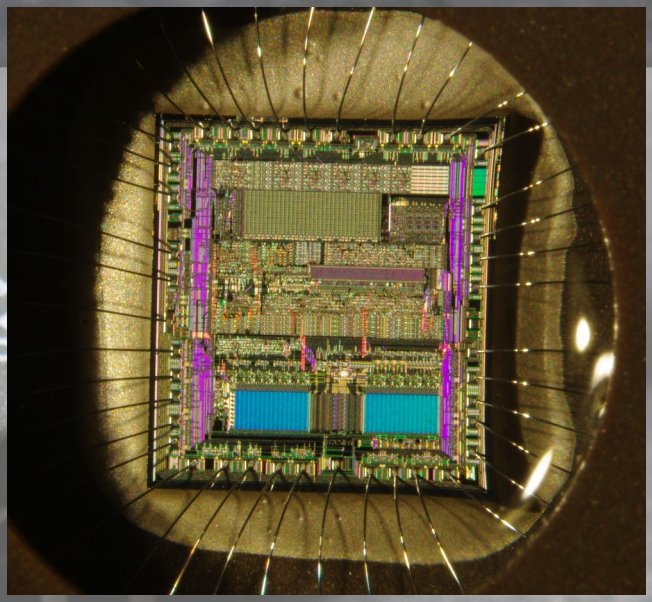
Goes into ADC

$$V_{out} = V_{CC} \times \left(\frac{R_x}{R_3 + R_x} - \frac{R_2}{R_1 + R_2} \right)$$

Processing subsystem

- The processor subsystem
  - interconnects all the other subsystems and some additional peripheries
  - its main purpose is to execute instructions pertaining to sensing, communication, and self-organization
- It consists of
  - processor chip
  - nonvolatile memory - stores program instructions
  - active memory - temporarily stores the sensed data
  - internal clock
- Usually implemented using microcontrollers, but other possibilities exist:
  - DSPs
  - FPGAs
  - ASICs

Microcontroller



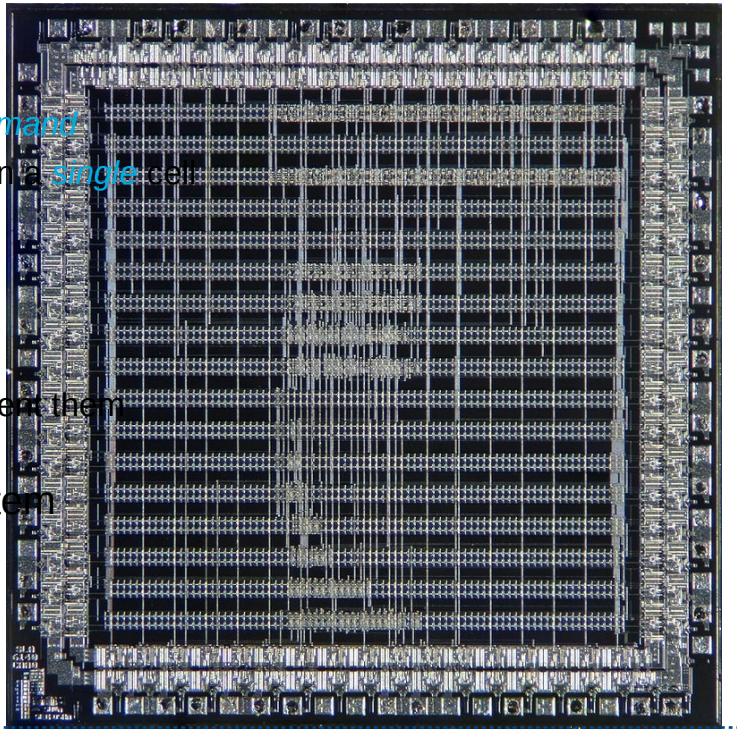
- Integrates:
  - CPU core
  - volatile memory (RAM) for data storage
  - ROM, EPROM, EEPROM, or Flash memory
  - parallel I/O interfaces
  - discrete input and output bits
  - clock generator
  - one or more internal analog-to-digital converters
  - serial communications interfaces
- Advantages:
  - suitable for building computationally less intensive, standalone applications, because of its *compact construction*, *small size*, *low-power consumption*, and *low cost*
  - *high speed* of the programming and eases debugging, because of the use of higher-level programming languages
- Disadvantages:
  - *not* as *powerful* and as *efficient* as some custom-made processors (such as DSPs and FPGAs)
  - some applications (simple sensing tasks but large scale deployments) may prefer to use architecturally simple but energy- and cost-efficient processors

Digital Signal Processor

- The main function:
  - process discrete signals with digital filters
  - filters minimize the effect of noise on a signal or enhance or modify the spectral characteristics of a signal while analog signal processing
  - requires complex hardware components, digital signal processors (DSP) requires simple adders, multipliers, and delay circuits
  - DSPs are highly efficient
- *Advantages:*
  - *powerful* and *complex* digital filters can be realized with *commonplace* DSPs
  - *useful for applications* that require the deployment of nodes in harsh physical settings (where the signal transmission suffers corruption due to noise and interference and, hence, requires aggressive signal processing)
- *Disadvantage:*
  - some tasks require *protocols* (and not numerical operations) that *require* periodical *upgrades* or *modifications* (i.e., the networks should support flexibility in network reprogramming)

Application-specific Integrated Circuit

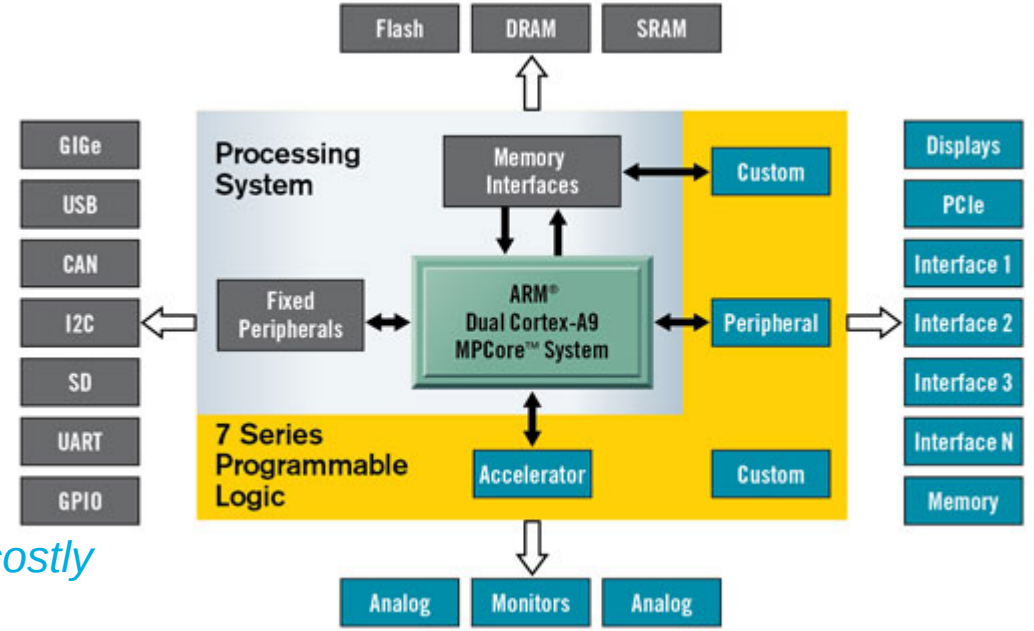
- ASIC is an *IC* that can be customized for a specific application
- Two types of design approaches: *full-customized* and *half-customized*
  - full-customized IC:
    - some logic cells, circuits, or layout are custom made in order to optimize cell performance
    - includes features which are not defined by the standard cell library
    - expensive and long design time
  - half-customized ASICs are built with logic cells that are available in the standard library
  - in both cases, the final logic structure is configured by the end user - an ASIC is a *cost efficient solution*, *flexible*, and *reusable*
- *Advantages:*
  - relatively *simple design*; can be optimized to *meet a specific customer demand*
  - *multiple* microprocessor *cores* and *embedded software* can be designed in a *single* cell
- *Disadvantage:*
  - *high* development *costs* and *lack of re-configurability*
- *Application:*
  - ASICs are not meant to replace microcontrollers or DSPs but to complement them
  - handle rudimentary and low-level tasks
    - to decouple these tasks from the main processing subsystem



Microscope photograph of a gate-array ASIC showing the predefined logic cells and custom interconnections. This particular design uses less than 20% of available logic gates.

Field Programmable Gate Array (FPGA)

- The distinction between ASICs and FPGAs is not always clear
  - FPGAs are *more complex* in design and *more flexible* to program
  - FPGAs are programmed electrically, by modifying a packaged part
  - programming is done with the support of circuit diagrams and hardware description languages, such as VHDL and Verilog
- *Advantages:*
  - *higher bandwidth* compared to DSPs
  - *flexible* in their application
  - support *parallel processing*
  - work with *floating point representation*
  - greater *flexibility of control*
- *Disadvantages:*
  - *Complex*
  - the design and realization process is *costly*



Processing Subsystem - Summary

- Working with a *micro-controller* is *preferred* if the design goal is to achieve *flexibility*
- Working with the other mentioned options is preferred if power consumption and computational efficiency is desired
- *DSPs* are expensive, large in size and less flexible; they are *best for signal processing*, with specific algorithms
- *FPGAs are faster* than both microcontrollers and digital signal processors and support *parallel computing*; but their production cost and the programming difficulty make them *less suitable*
- *ASICs have higher bandwidths; they are the smallest in size, perform much better, and consume less power than any of the other processing types; but have a high cost of production owing to the complex design process*

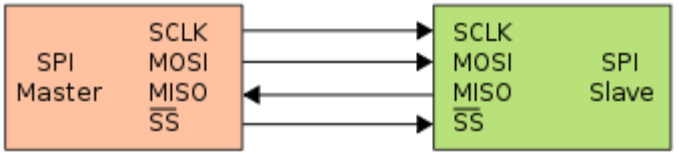
Communication Interfaces

- *Fast* and *energy efficient data transfer* between the subsystems of a wireless sensor node *is vital*
  - however, the practical size of the node puts restriction on system buses
  - communication via a parallel bus is *faster* than a serial transmission
  - a parallel bus needs *more space*
- Therefore, considering *the size of the node*, *parallel buses* are never supported
- The choice is often between *serial interfaces* :
  - Serial Peripheral Interface (SPI)
  - General Purpose Input/Output (GPIO)
  - Secure Data Input/Output (SDIO)
  - Inter-Integrated Circuit (I^2C)

Among these, the most commonly used buses are *SPI* and *I^2C*

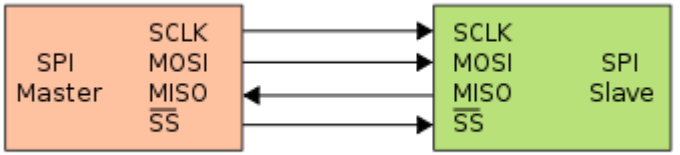
Serial Peripheral Interface

- SPI (Motorola, in the mid-80s)
  - *high-speed*, *full-duplex synchronous* serial bus
  - does *not* have an official standard, but use of the SPI interface should conform to the implementation specification of others - correct communication
- The SPI bus defines *four pins*:
  - MOSI (MasterOut/SlaveIn)
    - used to transmit data *from the master to the slave* when a device is configured as a master
    - MISO (MasterIn/SlaveOut)
    - SCLK (Serial Clock)
      - used by the *master* to send the clock signal that is needed to *synchronize transmission*
      - used by the *slave* to read this signal synchronize transmission
    - CS (Chip Select) - communicate via the CS port



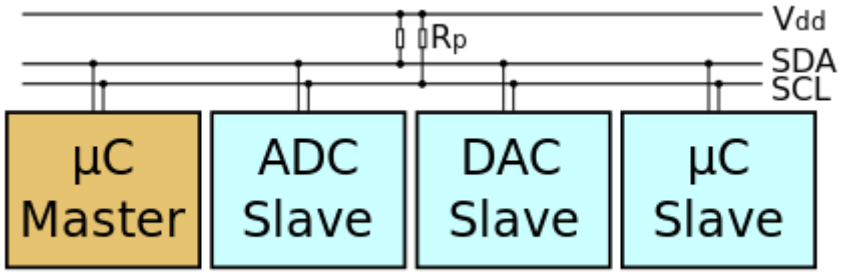
Serial Peripheral Interface

- Both master and slave devices hold a shift register
- Every device in every transmission must read and send data
- SPI supports a *synchronous communication protocol*
  - the master and the slave must agree on the timing
  - master and slave should agree on two additional parameters
    - clock polarity (*CPOL*) - defines whether a clock is used as high- or low-active
    - clock phase (*CPHA*) - determines the times when the data in the registers is allowed to change and when the written data can be read



Inter-Integrated Circuit

- Every device type that uses I²C must have a unique address that will be used to communicate with a device
- In earlier versions, a *7 bit address* was used, allowing 112 devices to be uniquely addressed - due to an increasing number of devices, it *is insufficient*
- Currently I²C uses *10 bit addressing*
- I²C is a *multi-master half-duplex synchronous serial* bus
  - only two bidirectional lines: (unlike SPI, which uses four)
    - Serial Clock (SCL)
    - Serial Data (SDA)



Inter-Integrated Circuit

- Since each master generates its own clock signal, communicating devices must *synchronize their clock speeds*
  - a slower slave device could wrongly detect its address on the SDA line while a faster master device is sending data to a third device
- I²C requires arbitration between master devices *wanting* to send or receive data at the same time
  - *no* fair arbitration *algorithm*
  - rather the master that holds the SDA line low for *the longest time wins* the medium
- I²C enables a device to read data *at a byte level* for a fast communication
  - the device can hold the SCL low until it completes reading or sending the next byte - called *handshaking*
- The *aim* of I²C is *to minimize costs* for connecting devices
  - accommodating lower transmission speeds
- I²C defines two speed modes:
  - *a fast-mode* - a bit rate of up to 400Kbps
  - *high-speed mode* - a transmission rate of up to 3.2 Mbps
  - they are downwards compatible to ensure communication with older components

Communication Interfaces - Comparison

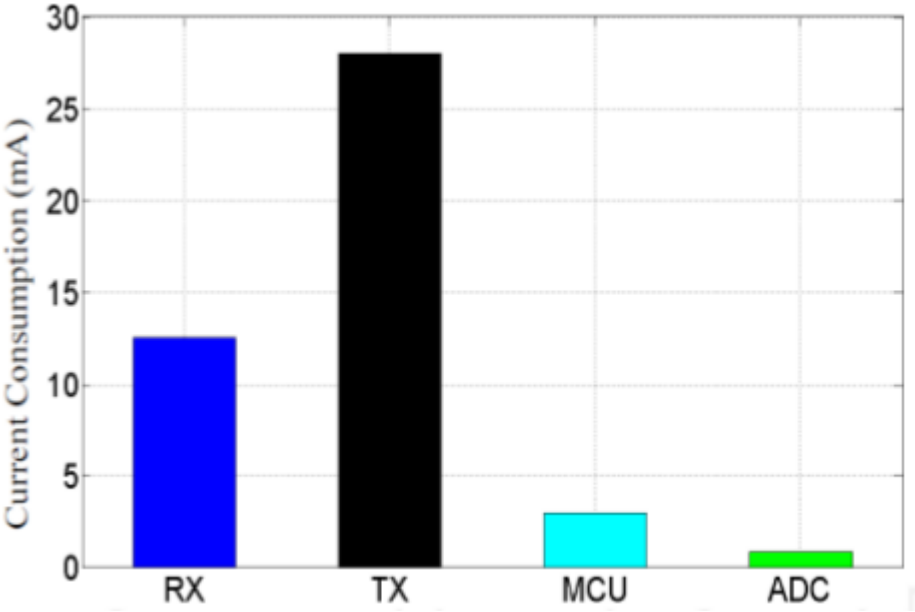
SPI	**I²C**
4 lines enable full-duplex transmission	2 lines reduce space and simplify circuit layout; Lowers costs
No addressing is required due to CS	Addressing enables multi-master mode; Arbitration is required
Allowing only one master avoids conflicts	Multi-master mode is prone to conflicts
Hardware requirement support increases with an increasing number of connected devices -- costly	Hardware requirement is independent of the number of devices using the bus
The master's clock is configured according to the slave's speed but speed adaptation slows down the master.	Slower devices may stretch the clock -- latency but keeping other devices waiting
Speed depends on the maximum speed of the slowest device	Speed is limited to 3.2 Mbps
Heterogeneous registers size allows flexibility in the devices that are supported.	Homogeneous register size reduces overhead
Combined registers imply every transmission should be read AND write	Devices that do not read or provide data are not forced to provide potentially useless bytes
The absence of an official standard leads to application specific implementations	Official standard eases integration of devices since developers can rely on a certain implementation

Communication Interfaces - Summary

- Buses are essential highways to transfer data
  - due to the concern for size, only serial buses can be used
  - serial buses demand high clock speeds to gain the same throughput as parallel buses
  - serial buses can also be bottlenecks (e.g., Von Neumann architecture) or may not scale well with processor speed (e.g., I^2C)
- Delays due to contention for bus access become critical, for example, if some of the devices act unfairly and keep the bus occupied

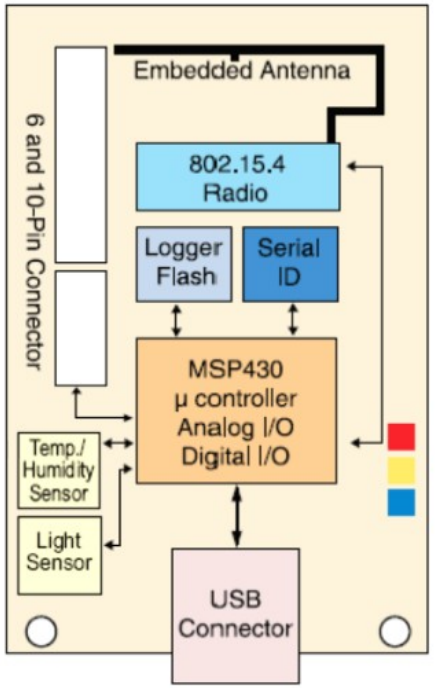
Wireless Transceivers

- Transceivers
  - Conventional: low-level PHY functionalities:
    - frequency and channels, spectrum handling,
    - modulation, bit rate. Advanced network functionalities and processing are implemented on software (i.e. microprocessor)
  - Current Trend:
    - System-on-Chip -> allows implementation of a sophisticated protocol stack on the chip (dedicated microprocessor & memory)
    - Either way: it is the element with the highest power consumption
      - Radio Duty Cycling: putting transceiver to different states:
        - Transmit / Receive
        - Idle: ready to receive
        - Sleep: significant parts of the chip are switched off



TelosB Node

Module		
Processor Performance	16-bit RISC	
Program Flash Memory	48K bytes	
Measurement Serial Flash	1024K bytes	
RAM	10K bytes	
Configuration EEPROM	16K bytes	
Serial Communications	UART	0-3V transmission levels
Analog to Digital Converter	12 bit ADC	8 channels, 0-3V input
Digital to Analog Converter	12 bit DAC	2 ports
Other Interfaces	Digital I/O,I2C,SPI	
Current Draw	1.8 mA	Active mode
	5.1 µA	Sleep mode
RF Transceiver		
Frequency band[1]	2400 MHz to 2483.5 MHz	ISM band
Transmit (TX) data rate	250 kbps	
RF power	-24 dBm to 0 dBm	
Receive Sensitivity	-90 dBm (min), -94 dBm (typ)	
Adjacent channel rejection	47 dB	+ 5 MHz channel spacing
	38 dB	- 5 MHz channel spacing
Outdoor Range	75 m to 100 m	Inverted-F antenna
Indoor Range	20 m to 30 m	Inverted-F antenna
Current Draw	23 mA	Receive mode
	21 µA	Idle mode
	1 µA	Sleep mode
Sensors		
Visible Light Sensor Range	320 nm to 730 nm	Hamamatsu S1087
Visible to IR Sensor Range	320 nm to 1100nm	Hamamatsu S1087-01
Humidity Sensor Range	0-100% RH	Sensirion SHT11
Resolution	0.03% RH	
Accuracy	± 3.5% RH	Absolute RH
Temperature Sensor Range	-40°C to 123.8°C	Sensirion SHT11
Resolution	0.01°C	
Accuracy	± 0.5°C	@25°C
Electromechanical		
Battery	2X AA batteries	Attached pack
User Interface	USB	v1.1 or higher
Size (in)	2.55 x 1.24 x 0.24	Excluding battery pack
(mm)	65 x 31 x 6	Excluding battery pack
Weight (oz)	0.8	Excluding batteries
(grams)	23	Excluding batteries



Operating Systems

WSN Operating Systems

- An operating System is
  - a thin software layer
  - resides between the hardware and the application layer
  - provides basic programming abstractions to application developers
- Its *main task* is to enable applications to interact with hardware resources, to schedule and prioritize tasks.
- Operating systems are classified as: single-task/multitasking and single-user/multiuser operating systems
  - multi-tasking OS - the overhead of concurrent processing because of the limited resources
  - single task OS - tasks should have a short duration
- The choice of a particular OS depends on several factors; typically *functional* and *non-functional* aspects.
  - Here we focus on functional aspects.

Scheduling

- Two scheduling mechanisms:
  - queuing-based scheduling
    - FIFO - the simplest and has minimum system overhead, but treats tasks unfairly
    - sorted queue - e.g., shortest job first (SJF) - incurs system overhead (to estimate execution duration)
  - round-robin scheduling
    - a time sharing scheduling technique
    - several tasks can be processed concurrently
- Regarding how tasks are executed, a scheduler can be either
  - a non-preemptive scheduler - a task is executed to the end, may not be interrupted by another task
  - or preemptive scheduler - a task of higher priority may interrupt a task of low priority

Stacks & System Calls

- *Stacks*
  - a data structure that temporarily stores data objects in memory by piling one upon another
  - objects are accessed using last-in-first-out (LIFO)
- *System Calls*
  - decouple the concern of accessing hardware resources from implementation details
  - whenever users wish to access a hardware resource, they invoke these operations without the need to concern themselves how the hardware is accessed

Handling Interrupts

- An interrupt is an asynchronous signal generated by
  - a hardware device
  - several system events
  - OS itself
- An interrupt causes:
  - the processor to interrupt executing the present instruction
  - to call for an appropriate interrupt handler
- Interrupt signals can have different priority levels, a high priority interrupt can interrupt a low level interrupt
- Interrupt mask: let programs choose whether or not they wish to be interrupted

Multi-threading

- A *thread* is the path taken by a processor or a program during its execution
- *Multi-threading* - a task is divided into several logical pieces
  - scheduled independent from each other
  - executed concurrently
- Two advantages of a multi-threaded OS:
  - tasks do not block other tasks
  - short-duration tasks can be executed along with long-duration tasks
- Threads cannot be created endlessly
  - the creation of threads *slows down* the processor
  - no sufficient resources to divide
- The OS can keep the number of threads to a *manageable size* using a thread pool

Thread-based vs. Event-based Programming

- Decision whether to use threads or events programming:
  - need for separate stacks
  - need to estimate maximum size for saving context information
- *Thread-based programs* use multiple threads of control within:
  - a single program
  - a single address space
  - *Advantage:*
    - a thread blocked can be suspended while other tasks are executed in different threads
  - *Disadvantages:*
    - must carefully protect shared data structures with locks
    - use condition variables to coordinate the execution of threads
- *event-based programming*: use events and event handlers
  - event-handlers register with the OS scheduler to be notified when a named event occurs
  - a loop function:
    - polls for events
    - calls the appropriate event-handlers when events occur
  - An event is processed to completion
    - unless its handler reaches at a blocking operation (callback and returns control to the scheduler)

Memory Allocation

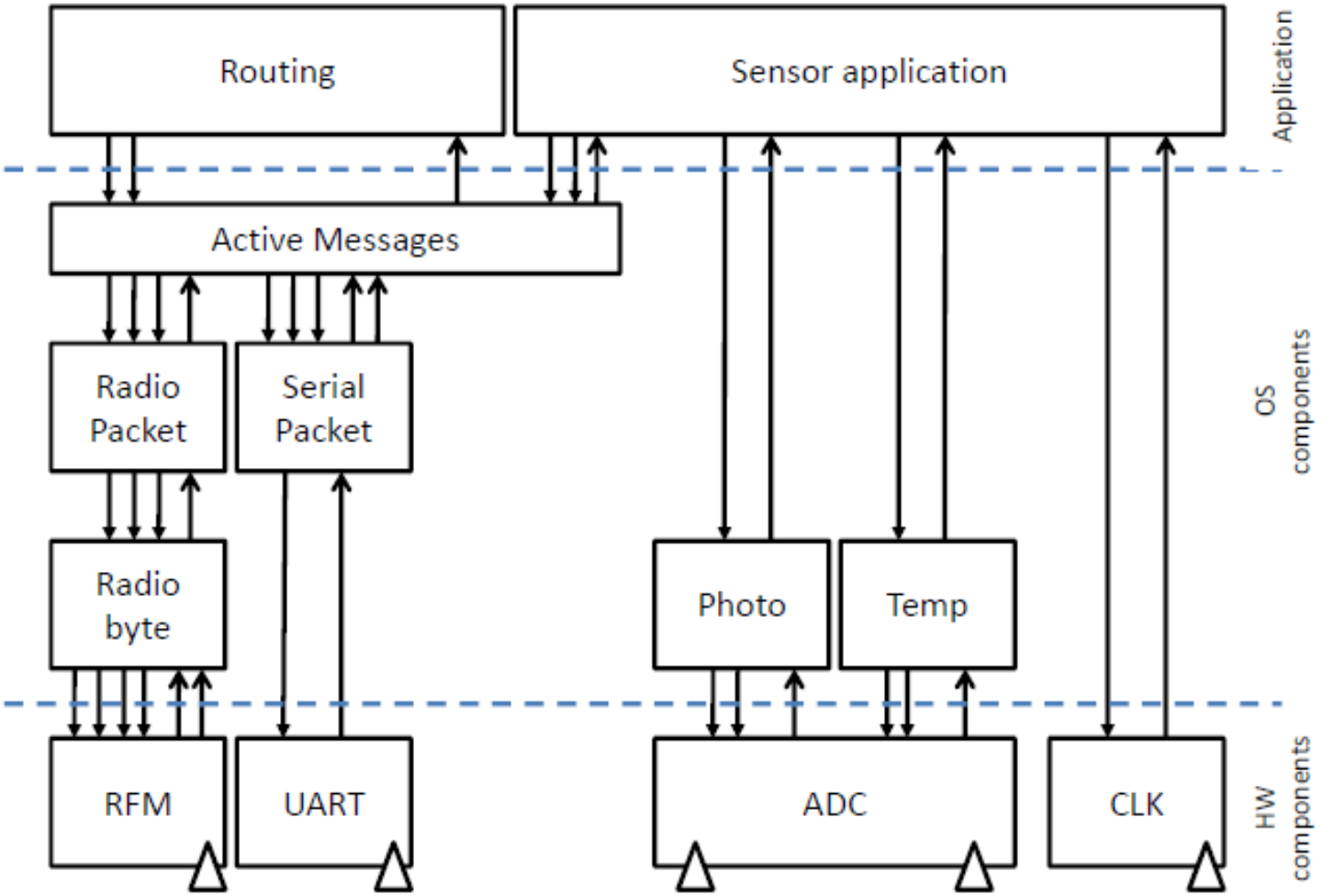
- The memory unit is a precious resource
- Reading and writing to memory is costly
- How and for how long a memory is allocated for a piece of program determines the speed of task execution
- Memory can be allocated to a program:
  - *statically* - a frugal approach, but the requirement of memory *must be known* in advance
    - memory is used efficiently
    - runtime adaptation is not allowed
  - *dynamically* - the requirement of memory is *not known* in advance (on a transient basis)
    - enables flexibility in programming
    - but produces a considerable management overhead

TinyOS (2000)

- TinyOS is *the most widely used*, *richly documented*, and *tool-assisted* runtime environment in WSN (perhaps this is changing now...)
  - static memory allocation
  - event-based system
- TinyOS's architecture consists of
  - a scheduler
  - a set of components, which are classified into
    - configuration components - "wiring" (how models are connected with each other)
    - modules - the basic building blocks of a TinyOS program
- A component is made up of
  - a frame
  - command handlers
  - event handlers
  - a set of non-preemptive tasks
- A component is similar to an object in object-based programming languages:
  - it encapsulates state and interacts through well-defined interfaces
  - an interface that can define commands, event handlers, and tasks

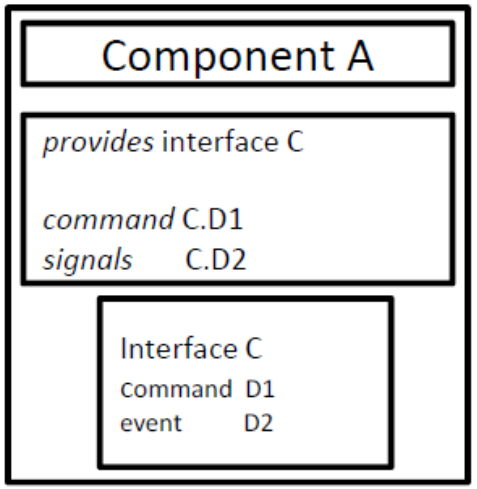
two components at the highest level communicate asynchronously through active messages

- routing component - establishing and maintaining the network
- sensor application - responsible for sensing and processing

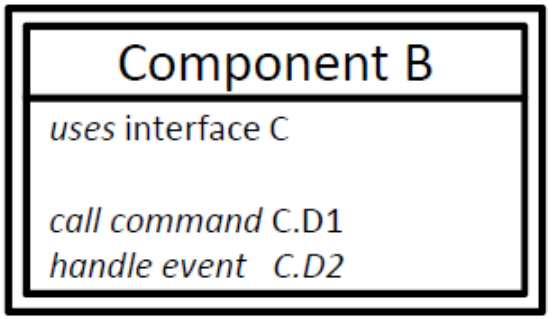


Binding components

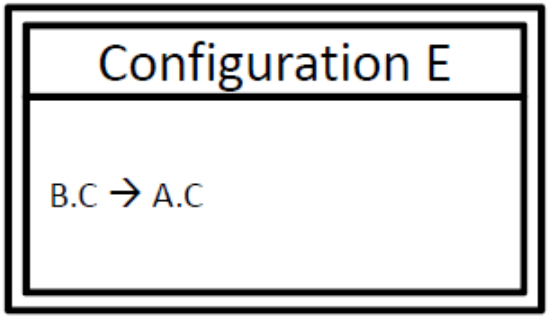
The logical structure of components and component configurations



A TinyOS component providing an interface



A TinyOS components that uses an interface



A TinyOS configuration that wires an interface provider and an interface user

Component A declares its service by providing *interface C*, which in turn provides *command D1* and signals *event D2*.

Component B expresses interest in *interface C* by declaring a call to *command D1* and by providing an event handler to process *event D2*.

a binding between *Component A* and *Component B* is established through the *Configuration E*.

Tasks, Commands and Events

- The fundamental building blocks of a TinyOS runtime environment: *tasks, commands,* and *events*
  - enabling effective communication between the components of a single frame
- *Tasks :*
  - *monolithic processes* - should execute to completion - they cannot be preempted by other tasks, though they can be interrupted by events
  - possible to allocate a single stack to store context information
  - call lower level commands; signal higher level events; and post (schedule) other tasks
  - scheduled based on FIFO principle (in TinyOS)

Tasks, Commands and Events

- *Commands:*
  - non-blocking requests made by higher-level components to lower-level components
  - split-phase operation:
    - a function call returns immediately
    - the called function notifies the caller when the task is completed
- *Events:*
  - events are processed by the event handler
    - event handlers are called when hardware events occur
    - an event handler may react to the occurrence of an event in different ways
      - deposit information into its frame, post tasks, signal higher level events, or call lower level commands

WSN Operating Systems

- Operating systems that implement the OpenWSN stack. (Watteyne et. al, 2016)

THE FOUR OPERATING SYSTEMS IMPLEMENTING THE 6TISCH STACK STUDIED IN THIS PAPER.

Name	Programming Model	Targeted Devices[1]	Supported MCU Families or Vendors	Developed Since	Supported RPL Modes	6LBR Implementation
OpenWSN	event-driven	Class 0 – 2	MSP430, ARM Cortex-M	2010	non-storing	on the host
Contiki	event-driven, protothreads	Class 0 – 2	AVR, MSP430, PIC32, ARM CM, OpenRISC, ARM7, x86	2002	storing	on the mote or the host
RIOT	multi-threading	Class 1 + 2	AVR, MSP430, ARM7, ARM Cortex-M, x86	2012	storing and non-storing	on the mote
TinyOS	event-driven	Class 0 – 2	AVR, MSP430, px27ax, ARM Cortex-M, OpenRISC	2000	storing	on the host

[1] According to RFC7228 [32]: Class 0 devices have $\ll 10kB$ RAM and $\ll 100kB$ ROM, Class 1 devices have $\sim 10kB$ and $\sim 100kB$ ROM, Class 2 devices have $\sim 50kB$ and $\sim 250kB$ ROM.

Assigment

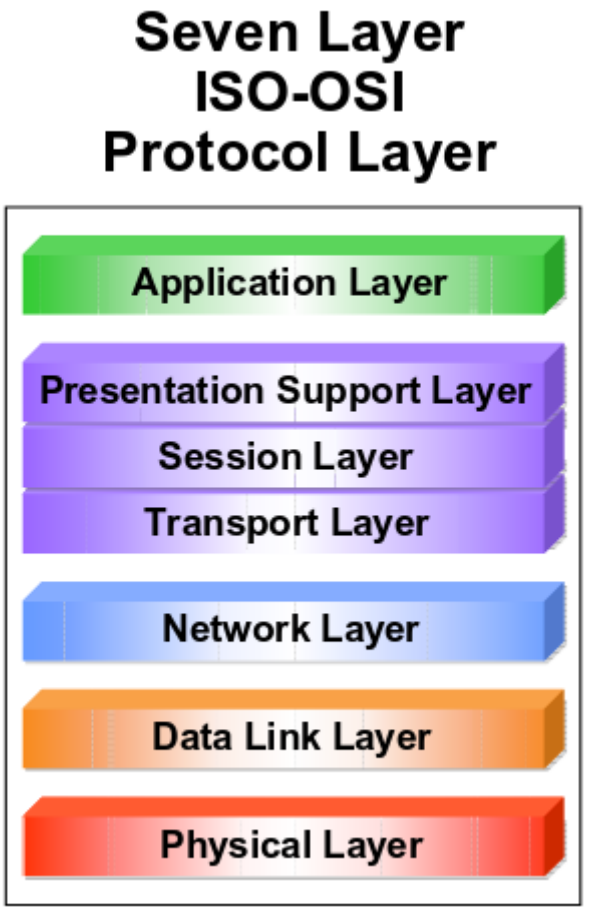
- Implement a simple application with timing constraints using:
  - OpenWSN stack (TSCH + 6TiSCH stack)
  - RIOT OS or OpenWSN (FreeRTOS based)
  - HW: TelosB WSN node
- Start by installing the OS and compiling/running a simple example application.
- We will talk about OpenWSN and WSN communication basics in the next lecture.

Communication Architectures

OSI Reference Model

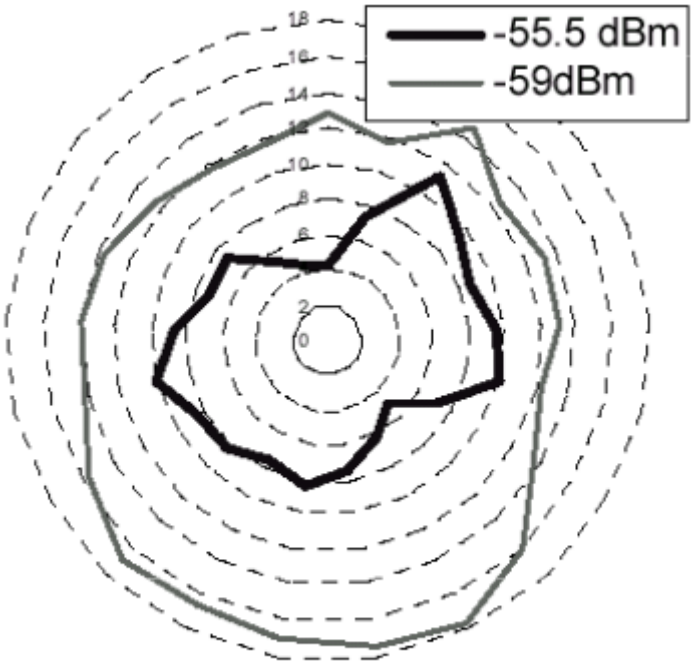
- Open Systems Interconnection model
  - standardizes the communication functions for:
    - Comunication systems
    - Computing systems
  - More layers - higher complexity – higher computational and memory demands
  - But... more scalable and increased interoperability
    - Fundamental in standardization efforts!

  Protocol Stack Model for WSN
  - Physical Layer: (de)modulation, spectrum allocation, transmission and reception (relying on well defined techniques and standards)
  - Data Link Layer (noisy environment / dynamic topologies) Error Control techniques for reliable communication and manage channel access through the MAC sublayer
  - Network Layer: (Energy-aware) data routing
  - Transport layer: Data flow maintenance



Physical Layer aspects

- radio link characteristics
  - Link asymmetry
    - node A is connected to Node B does not mean that Node B is connected to node A
  - non-isotropic connectivity
    - connectivity depends on the direction of the signal (at same distance from source)
  - non-monotonic distance decay
    - nodes geographically far away from source may get better connectivity than nodes that are geographically closer



*Zhou et. al. 04

Physical Layer aspects

- propagation phenomena
  - Reflection
    - is the change in direction of a wa between two different media so t returns into the medium from wh
  - Diffraction
    - refer to various phenomena whic encounters an obstacle
  - Scattering
    - from objects that are small (wher wavelength), e.g.: rough surface

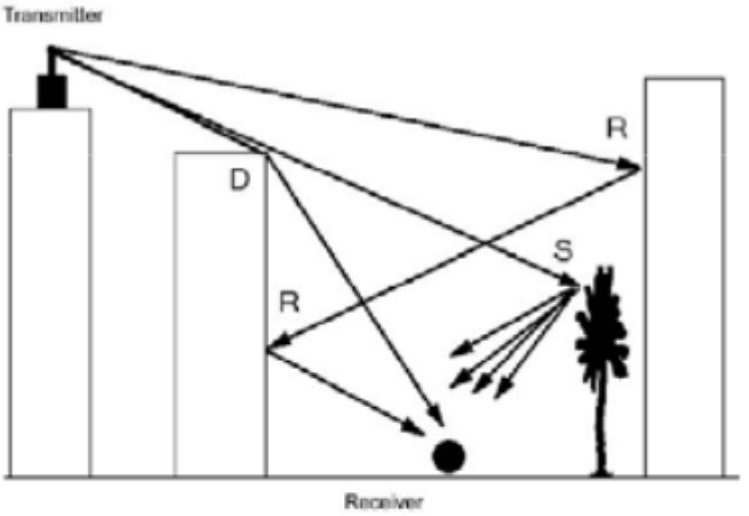
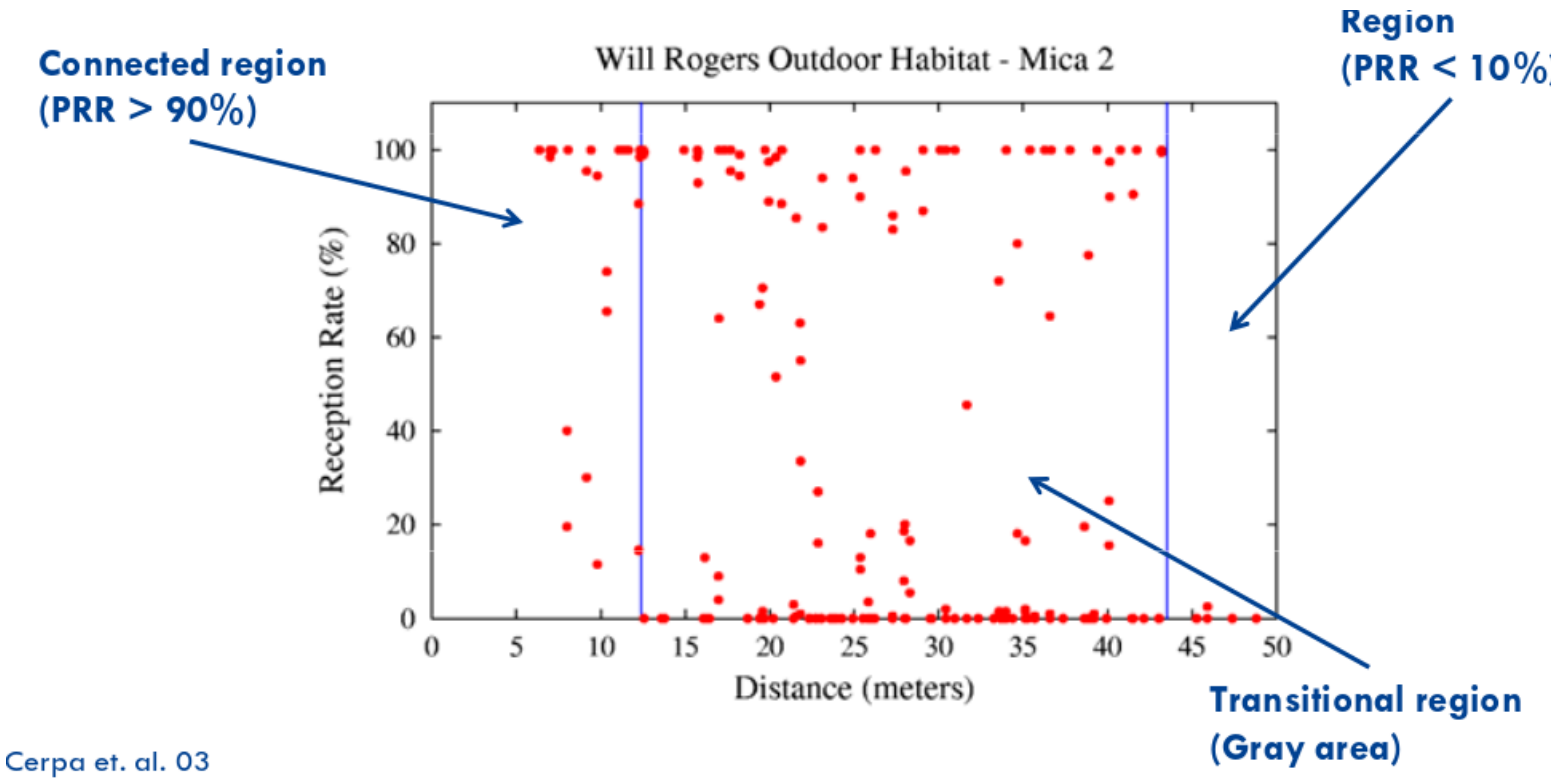


Figure 2.5 Reflection (R), diffraction (D) and scattering (S)

Source: **Wireless Networks,**
P. Nicopolitidis, A. S. Pomportsis,
G. I. Papadimitriou, M. S. Obaidat
Publisher John Wiley & Sons, Inc. New York, NY, USA (2003)

Physical Layer aspects

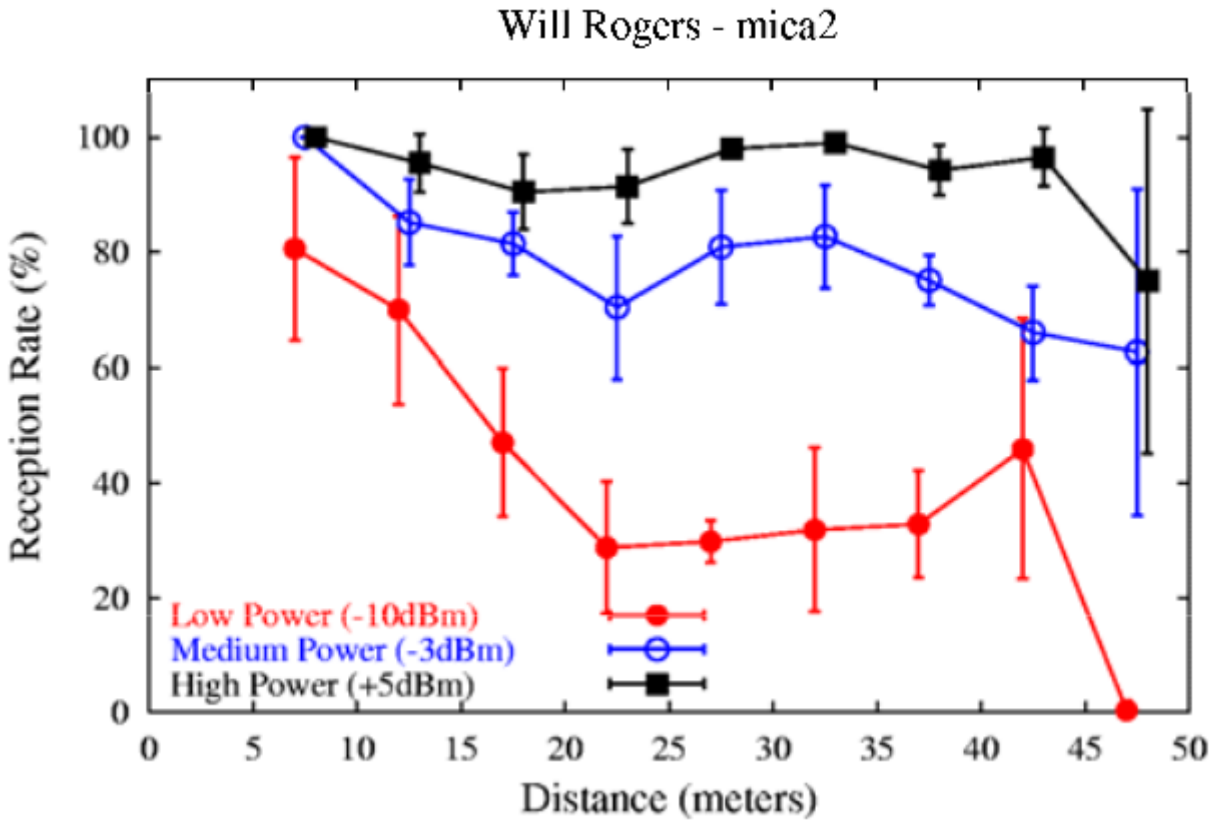
- spatial characteristics



Cerpa et. al. 03

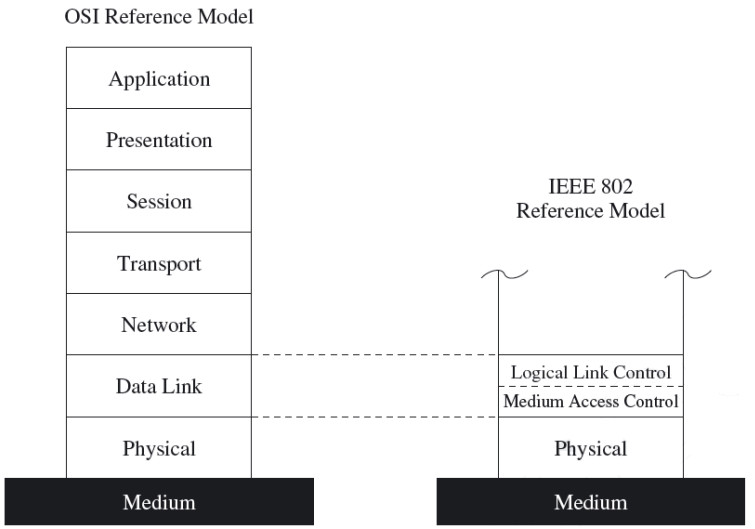
Physical Layer aspects

- packet reception rate vs. distance



Data Link Layer

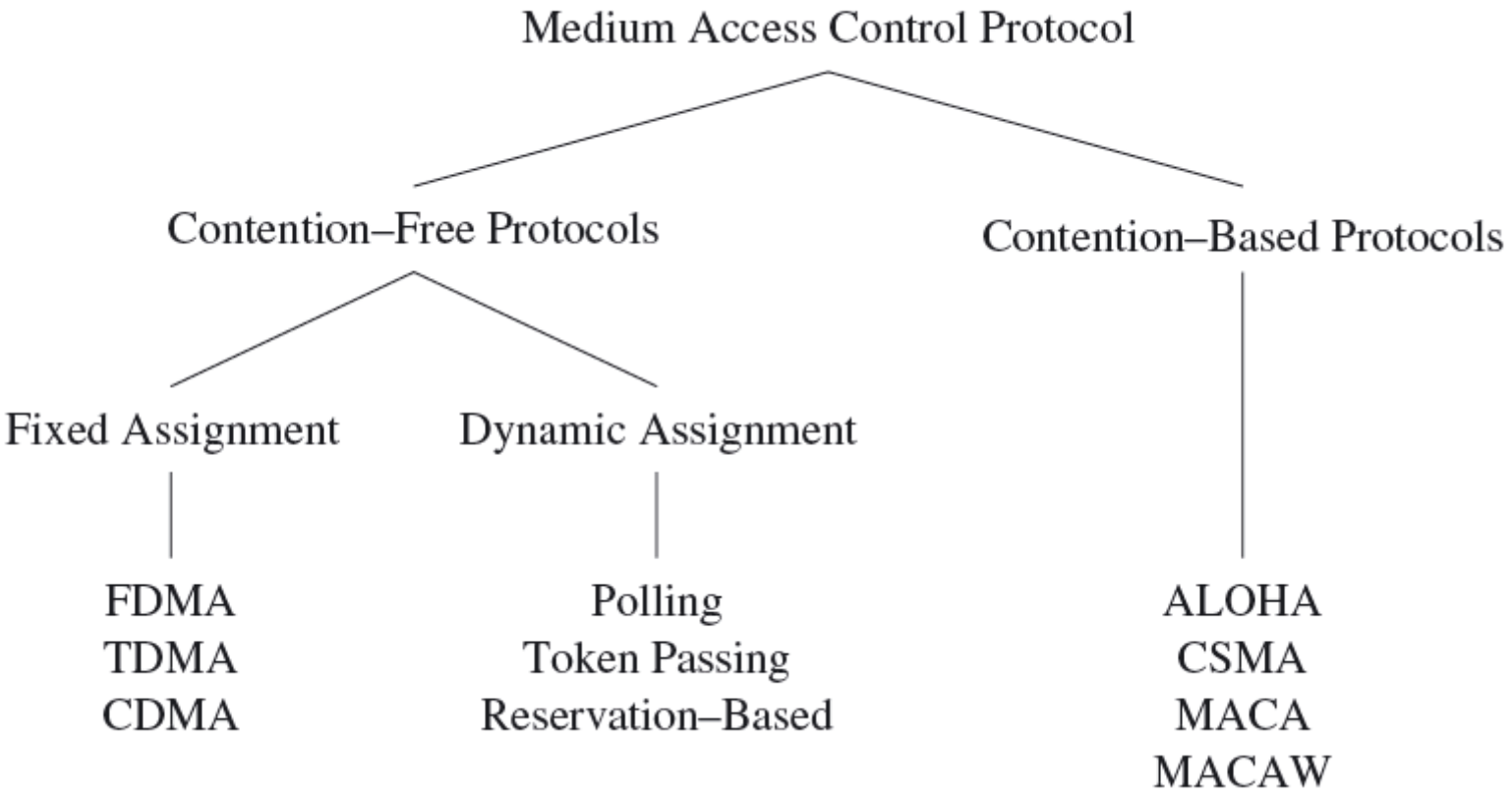
- In most networks, multiple nodes share a communication medium for transmitting their data packets
- The medium access control (MAC) protocol is primarily responsible for regulating access to the shared medium
- The choice of MAC protocol has a direct bearing on the reliability and efficiency of network transmissions
  - due to errors and interferences in wireless communications and to other challenges
- Energy efficiency also affects the protocol
  - trade energy efficiency for increase throughput or fairness



MAC Sub-layer

- Responsibilities of MAC layer include:
  - decide when a node accesses a shared medium
  - resolve any potential conflicts between competing nodes
  - correct communication errors occurring at the physical layer
  - perform other activities such as framing, addressing, and flow control

MAC Sub-layer



Contention-Free Medium Access

- Collisions can be avoided by ensuring that each node can use its allocated resources exclusively
- Examples of fixed assignment strategies:
  - FDMA: Frequency Division Multiple Access
    - the frequency band is divided into several smaller frequency bands
    - the data transfer between a pair of nodes uses one frequency band
    - all other nodes use a different frequency band
  - TDMA: Time Division Multiple Access
    - multiple devices to use the same frequency band
    - relies on periodic time windows (frames)
      - frames consist of a fixed number of transmission slots to separate the medium accesses of different devices
    - a time schedule indicates which node may transmit data during a certain slot

Contention-Free Medium Access

- Examples of fixed assignment strategies (contd.):
  - CDMA: Code Division Multiple Access
    - simultaneous accesses of the wireless medium are supported using different codes
    - if these codes are orthogonal, it is possible for multiple communications to share the same frequency band
    - forward error correction (FEC) at the receiver is used to recover from interferences among these simultaneous communications
- Fixed assignment strategies are inefficient
  - it is impossible to reallocate slots belonging to one device to other devices if not needed in every frame
    - generating schedules for an entire network can be a taunting task
    - these schedules may require modifications every time the network topology or traffic characteristics in the network change

Contention-Free Medium Access

- Dynamic assignment strategies: allow nodes to access the medium on demand
  - polling-based protocols
    - a controller device issues small polling frames in a round-robin fashion, asking each station if it has data to send
    - if no data to be sent, the controller polls the next station
  - token passing
    - stations pass a polling request to each other (round-robin fashion) using a special frame called a token
    - a station is allowed to transmit data only when it holds the token
  - reservation-based protocols
    - static time slots used to reserve future access to the medium
    - e.g., a node can indicate its desire to transmit data by toggling a reservation bit in a fixed location
    - these often very complex protocols then ensure that other potentially conflicting nodes take note of such a reservation to avoid collisions

Contention-Based Medium Access

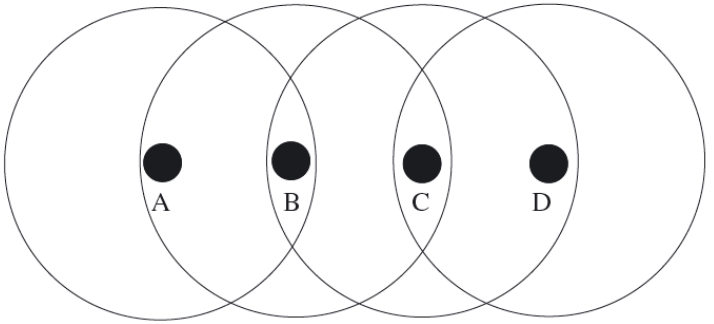
- Nodes may initiate transmissions at the same time
  - requires mechanisms to reduce the number of collisions and to recover from collisions
- Example 1: ALOHA protocol
  - uses acknowledgments to confirm the success of a broadcast data transmission
    - allows nodes to access the medium immediately
    - addresses collisions with approaches such as exponential back-off to increase the likelihood of successful transmissions
- Example 2: slotted-ALOHA protocol
  - requires that a station may commence transmission only at predefined points in time (the beginning of a time slot)
  - increases the efficiency of ALOHA
  - introduces the need for synchronization among nodes

Contention-Based Medium Access

- Destructive collisions
  - CSMA with Collision Avoidance (CSMA/CA)
    - nodes sense the medium, but do not immediately access the channel when it is found idle
    - instead, a node waits for a time period (Contention Window)
    - in case there are multiple nodes attempting to access the medium, the one with the shorter back-off period will win
- non-destructive collisions
  - resolve bus conflicts by using a bitwise arbitration; each node has a unique identifier (= priority); Wire acts like a logic AND (0 is dominant, 1 is recessive); transmit identifier bit by bit and hear the medium; if a node sends a ‘1’ but hears a ‘0’, he loses;
  - CAN (cars), HomePlug (domotics), WiDOM(wireless)
    - pros: deterministic, time and energy-efficient
    - cons: synchronization, short tx/rx turnaround time (or 2 transceivers); multiple broadcast domains

Hidden and Exposed Terminal Problems

- Hidden-terminal problem
  - senders A and C are able to reach B, but cannot overhear each other's signals
  - it is possible for A and C to transmit data to B at the same time, causing a collision at B, without being able to directly detect this collision
- Exposed-terminal problem
  - C wants to transmit data D, but decides to wait because it overhears an ongoing transmission from B to A
  - B's transmission could not interfere with data reception at C



Contention-Based Medium Access

- Multiple Access with Collision Avoidance (MACA)
  - dynamic reservation mechanism
  - sender indicates desire to send with ready-to-send (RTS) packet
  - intended receiver responds with clear-to-send (CTS) packet
  - if sender does not receive CTS, it will retry at later point in time
  - nodes overhearing RTS or CTS know that reservation has taken place and must wait (e.g., based on the size of data transmission)
  - address hidden terminal problem and reduces number of collisions
- MACA for Wireless LANs (MACAW)
  - receiver responds with acknowledgment (ACK) after data reception
    - other nodes in receiver's range learn that channel is available
  - nodes hearing RTS, but not CTS do not know if transmission will occur
    - MACAW uses data sending (DS) packet, sent by sender after receiving CTS to inform such nodes of successful handshake

MAC aspects - summary

- Characteristics of a good MAC/DLL protocol for WSNs
  - energy efficiency (to prolong the network lifetime)
    - flexible enough to adapt duty-cycles (100%→0%)
      - dynamically
      - in a per-cluster basis
    - must resolve some causes of energy loss:
      - Collisions (due to retransmissions)
      - Hidden-nodes and exposed-nodes(lead to unnecessary extra collisions)
      - Overhearing (wasted effort in receiving a packet destined to another node)
      - idle listening (sitting idly and trying to receive when nobody is sending)

MAC aspects - summary

- Characteristics of a good MAC/DLL protocol for WSNs (cont.)
  - scalability and adaptability
    - changes in network size node density and topology should be handled rapidly, transparently and effectively
  - Reliability
    - error detection/correction mechanisms; order inversion avoidance
  - traffic differentiation
    - support higher/lower priority traffic classes; support best-effort and real-time traffic
  - minimized frame overhead
    - but still support network management, security, error detection/correction