


Introduction to IoT

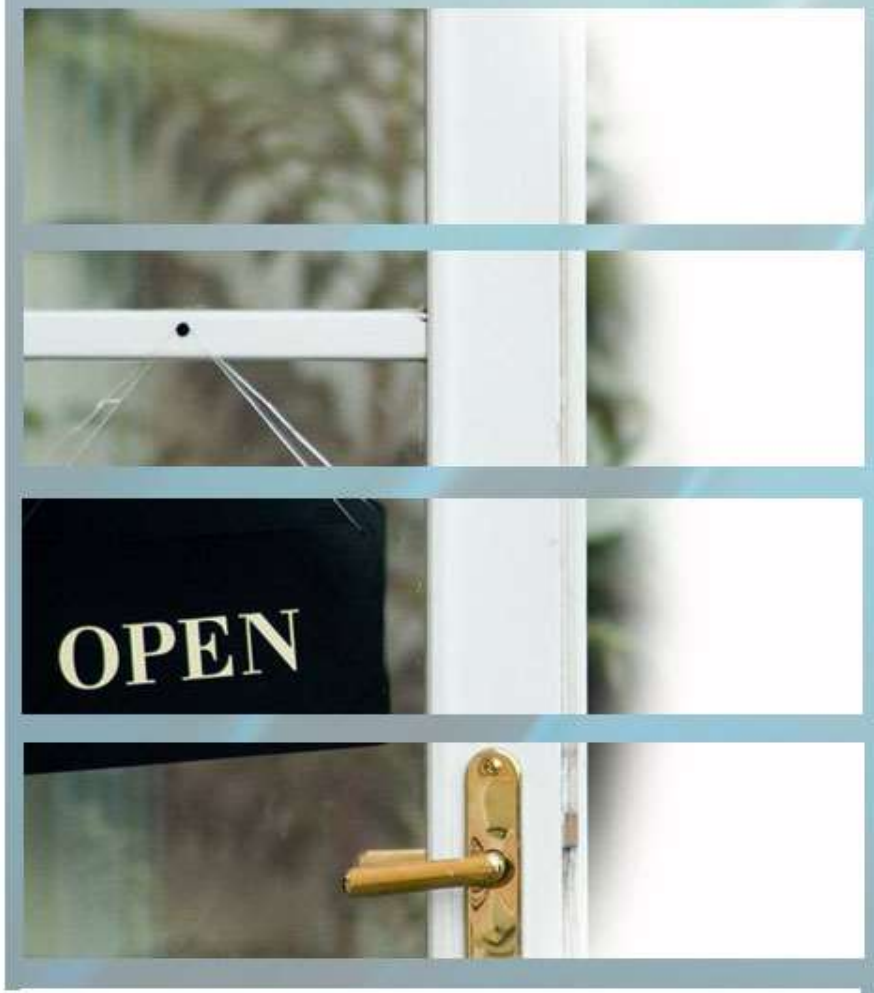
Jianhui Zhang, Ph.D., Prof.

College of Computer Science and Technology, Hangzhou Dianzi Univ.

Email: jh_zhang@hdu.edu.cn

Chapter Three

Sensor Technology

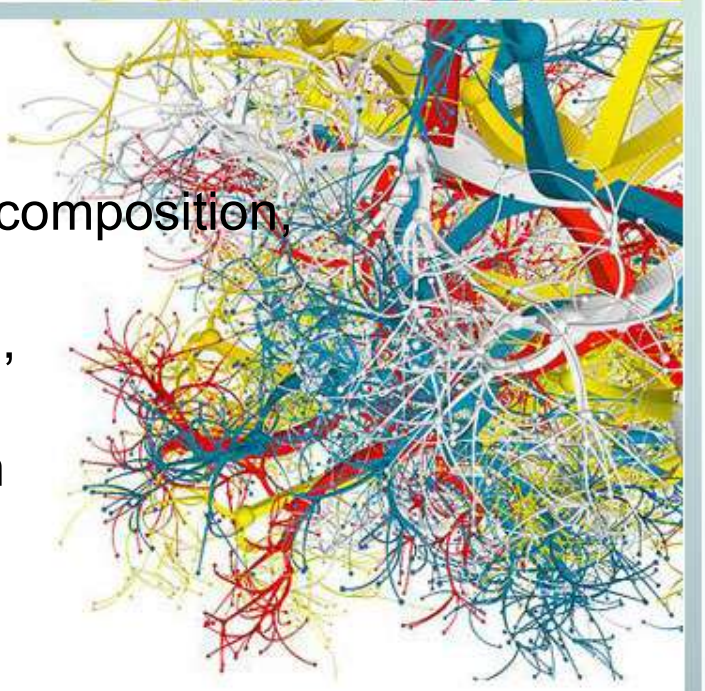


Sensors is an important means of information acquisition, together with communication technology and computer technology, constitute the three pillars of information technology.

This chapter introduces the development and application of sensors as well as hardware and software platforms.

Content

Review



Chapter 2 introduces the common automatic identification methods and technologies, focusing on RFID technology IC card system composition, one and two - dimensional barcode
The concept and system composition of RFID, the storage mode, classification and common frequency of tags RFID tag conflict prevention method (based on ALOHA protocol/based on binary tree protocol)

This chapter focuses on the introduction of sensor technology, involving the basic concepts and typical applications of sensors, as well as common hardware platforms and operating systems.

Content

3.1 Overview of sensors

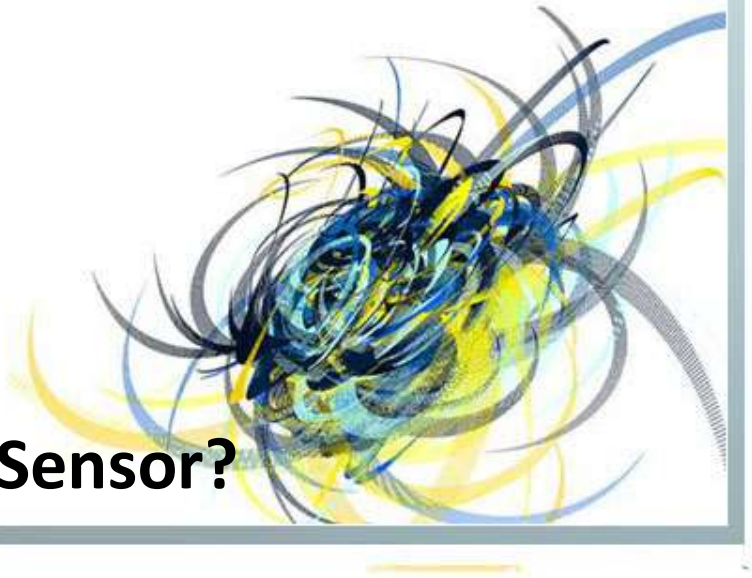
3.2 History of sensor technology

3.3 Typical applications

3.4 Design requirements

3.5 Hardware platform

3.6 Operating system



What is the Sensor?

What are the components of the Sensor?

3.1 Overview

Define

The national standard of China (GB7665-2005) defines a sensor as “A device or device that can sense the measured signal and convert it into usable output signal according to a certain rule".

Limitations of traditional sensors

Network, intelligent degree is very limited, lack of effective data processing and information sharing capacity

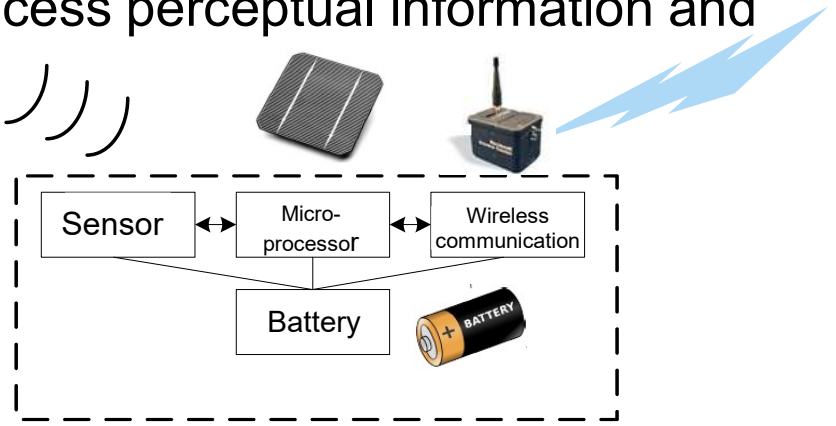
Modern sensor

Features: miniaturization, intelligence and networking

Typical representative: wireless sensor node

Wireless sensor node

Wireless sensor node composition: battery, sensor, microprocessor, wireless communication chip; Compared with traditional sensors, wireless sensor nodes not only include sensor components (pictured above on the left), but also integrate microprocessors and wireless communication chips, etc., which can analyze and process perceptual information and transmit it on the network.



Content

What are the two main lines of sensor development?

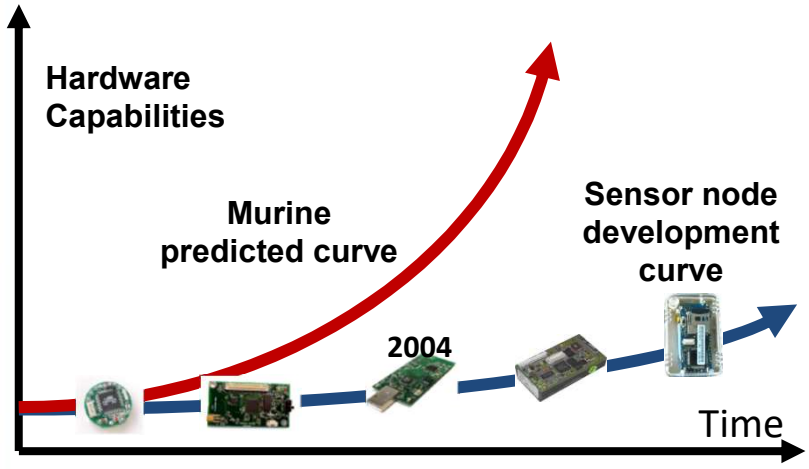
What are the constraints?

3.2 Development history of sensor technology: Two main lines

Start in the 1990s

- UCLA's LWIN program
  Low power wireless Sensor node
  - In 1996, the LWIM team developed the LWIM node by integrating multiple sensor, control, and communication chips into one device
  - In 1998, the LWIM team worked with Rockwell science center to develop WINS node
- The SmartDust program at Berkeley, California
  Miniaturized sensor node
  - In 1999, the university released the WeC node
  - The school then released a series of nodes, including Mica, Mica2, Mica2Dot, and MicaZ

3.2 Development history of sensor technology: Slowly improving performance



Computer hardware usually follows Moore's law: the number of transistors that can fit on an integrated circuit doubles every 18 months or so, doubling its performance.

Wireless sensor nodes are not growing as fast as Moore's law predicts!

What limits sensor performance?

Power consumption constraints: Wireless sensor nodes are generally deployed in the field, can not be wired power supply. Its hardware design must take energy saving as the important design goal.

Price constraints: Wireless sensor nodes generally need a large number of networks to complete specific functions. Its hardware must be designed to be cheap.

Size constraints: Wireless sensor nodes generally need to be easy to carry and easy to deploy. Its hardware design must be miniature as an important design goal.

Content

Although the sensor performance is limited, it is still widely used.

Sensors in military monitoring: VigilNet

VigilNet is a wireless sensor system developed by the university of Virginia for military surveillance. The system consists of XSM, Mica2, and Mica2Dot nodes with a maximum size of 200 nodes. The nodes are powered by batteries and laid alongside roads to detect and collect moving objects.

Application characteristics

- The nodes **independently form a network** and **transmit multiple hops**
- The nodes are **powered by batteries**, and the life cycle of the network is extended by **software energy-saving mechanism**
- Node **intelligent perception**, **cooperative work**, **upward warning** function

Sensors in smart buildings: LoCal

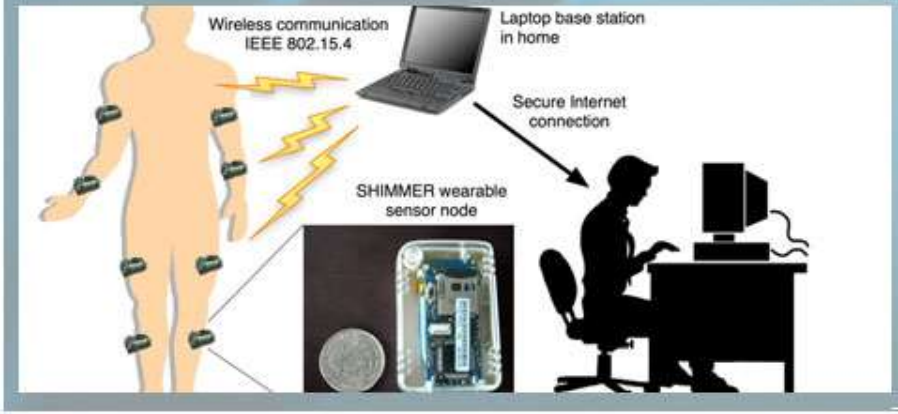
The annual American electricity report shows that at least 30% of electricity is wasted. Where does the electricity waste? Which of these could be saved? LoCal, a project led by the university of California, Berkeley, seeks to address these problems by deploying wireless sensor networks in smart buildings.

Application characteristics

- The sensor can realize **fine-grained perception** in space and time, and can track individual electrical appliances in real time
- Sensors can be "**multifunctional**" and can predict the user's behavior
- Sensors can be **interconnected**, and analysis of large amounts of continuous data can yield more useful information

Sensors in medical monitoring: Mercury

Another important application of sensors is medical monitoring. The Harvard team improved on traditional sensors, making them smaller and wearable



Application characteristics

- The design is very **humanization**
- The sensor has **high precision sensing ability**, and the medical data needs high sampling accuracy to be analyzed and diagnosed by doctors
- The sensor can collect data continuously for a long time
- The sensor USES **wireless communication**, and **its data transmission is opportunistic**

Content

Different application scenarios put forward unique design requirements for sensor hardware and software


Design requirements for large-scale, long-duration deployment of sensors

Low cost and Miniaturization

- Low-cost nodes can be deployed on a large scale, and miniaturized nodes can make deployment easier
- The software design of nodes also needs to meet the miniaturization requirements. For example, TelosB nodes have a memory size of only 4KB and program storage space of only 10KB.Therefore, the design of the node program must save computing resources and avoid exceeding the hardware capacity of the node

Design requirements for large-scale, long-duration deployment of sensors

Low power consumption

- **Low power chip is used in hardware design**

  For example, the TelosB node, uses a microprocessor with a power of 3mW, compared with 200 to 300W for a typical computer

- **Software energy saving strategy to achieve energy saving**

  The core of the software energy saving strategy is to try to make the node enter the low-power mode **when it does not need to work**, and only enter the normal state when it needs to work


Design requirements for large-scale, long-duration deployment of sensors

Flexibility and extensibility

- Sensor nodes are used in various applications, so the design of node hardware and software must be flexible and expandable
- The hardware design of nodes should meet certain standard interfaces. For example, the unified interface between nodes and sensor board is conducive to installing sensors with different functions on nodes
- The design of the software must be tailored so that different functional software modules can be installed according to the requirements of different applications

Design requirements for large-scale, long-duration deployment of sensors

Robustness

- Robustness is an important guarantee for long time deployment of sensor networks
- For ordinary computers, once the system crashes, people can restart the system, while sensor nodes cannot. For the whole network, redundancy can be appropriately increased to increase the robustness of the whole system

Content

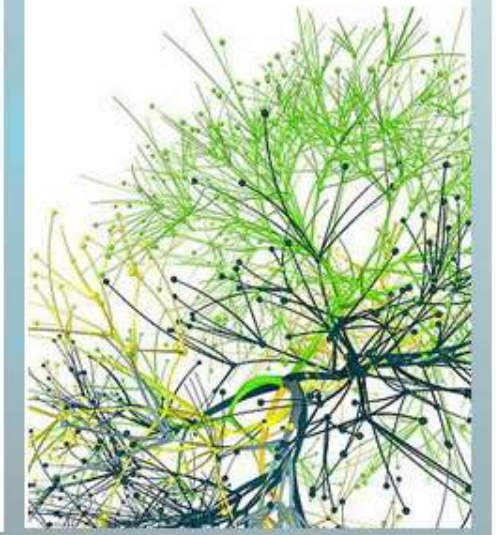
3.1 Overview of sensors
3.2 History of sensor technology
3.3 Typical applications
3.4 Design requirements
3.5 Hardware platform
3.6 Operating system

Combined with the design requirements, the basic characteristics of sensor node hardware platform can be obtained.

3.5 Hardware platform

Power device

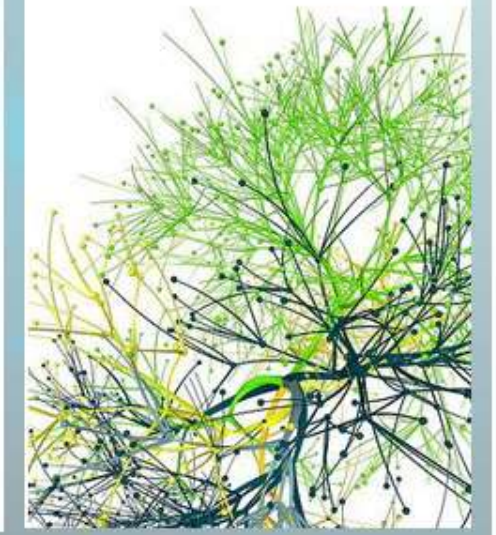
- Battery power makes the node easy to deploy. However, due to changes in voltage and environment, the battery capacity cannot be fully utilized.
- Renewable energy, such as solar energy. There are two ways to store energy from renewable energy sources: rechargeable batteries, which have less self-discharge and higher utilization of electricity, but lower charging efficiency and limited charging times; Ultracapacitor, high charging efficiency, charging times up to 1 million, and not easily affected by temperature, vibration and other factors.



3.5 Hardware platform

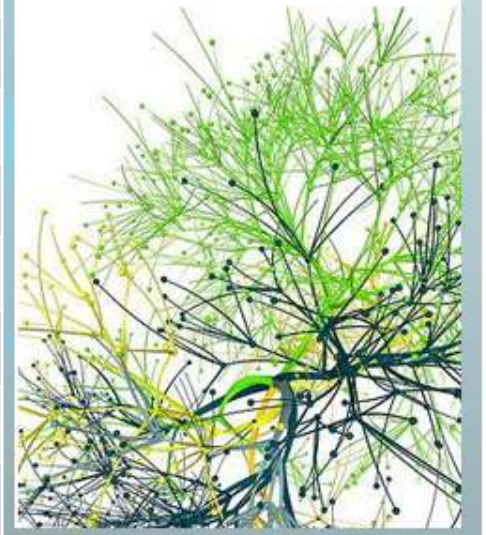
Sensor

There are many sensors available for node platform, which kind of sensor is often determined by the specific application requirements and the characteristics of the sensor itself.
Depending on how the processor interacts with the sensor: Select whether **external analog-to-digital converters** and **additional calibration techniques** are required, both through **analog signals** and through **digital signals**.



Common sensors and their key characteristics

Manufacturer	Sensor	Working voltage(V)	Energy consumption	Discrete sampling time
Taos	Visible light sensor	2.7-5.5	1.9mA	330us
Dallas Semiconductor	Temperature sensor	2.5-5.5	1mA	400ms
Sensirion	Humidity sensor	2.4-5.5	550uA	300ms
Intersema	Pressure sensor	2.4-3.6	1mA	35ms
Honeywell	Magnetic sensor	Any	4mA	30us
Analog Devices	Acceleration sensor	2.5-3.3	2mA	10ms
Panasonic	Sound sensor	2-10	0.5uA	1ms
Motorola	Smoke sensor	6-12	5uA	-
Melixis	Passive infrared sensor	Any	0mA	1ms
Li-Cor	Synthetic light sensor	Any	0mA	1ms
Ech2o	Soil moisture sensor	2-5	2mA	10ms

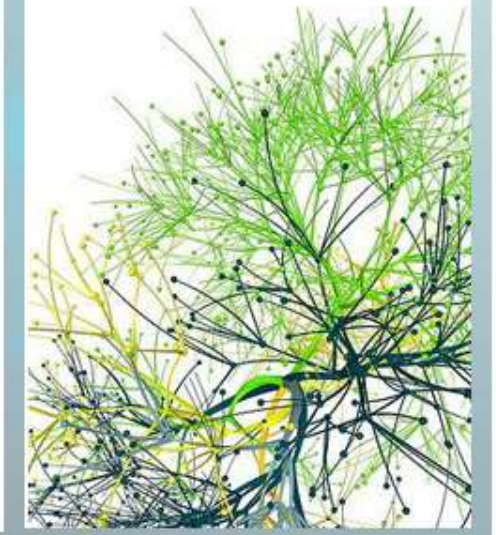


3.5 Hardware platform

The microprocessor

Microprocessor is the core of wireless sensor node **in charge of computing**. Current microprocessor chips also integrate **memory**, **flash memory**, **analog-to-digital converter**, **digital IO**, etc. Such **deep integration features** make them very suitable for use in wireless sensor network.

Key microprocessor performance factors that affect the overall performance of the node include **power consumption characteristics**, **wake time (fast switching between sleep and work), power supply voltage** (long working hours), **computing speed**, and **memory size**




Common microprocessors and their key features

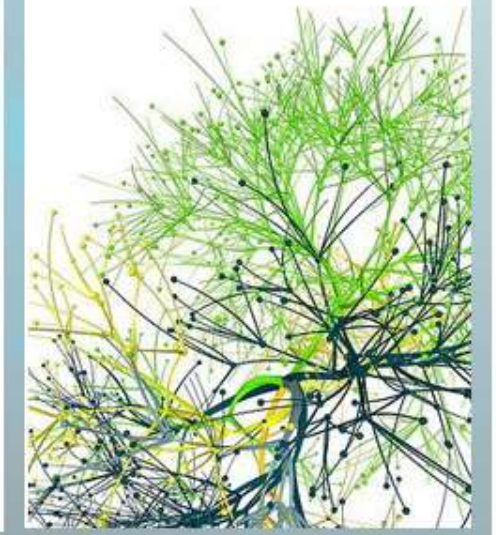
Manufacturer	Equipment	Year	Word Length(bit)	Working voltage	Memory(KB)	Flash Memory(KB)	Energy consumption during work(mA)	Energy consumption during sleep(uA)	Wake up time(us)
Atmel	Atmega128L	2002	8	2.7-5.5	4	128	0.95	5	6
	Atmega1281	2005	8	1.8-5.5	8	128	0.9	1	6
	Atmega1561	2005	8	1.8-5.6	8	256	0.9	1	6
Ember	EM250	2006	16	2.1-3.6	5	128	8.5	1.5	>1000
Freescale	HC05	1988	8	3.0-5.5	0.3	0	1	1	>2000
	HC08	1993	8	4.5-5.5	1	32	1	20	4
	HCS08	2003	8	2.7-5.5	4	60	7.4	1	10
Jennic	JN5121	2005	32	2.2-3.6	96	128	4.2	5	>2500
	JN5139	2007	32	2.2-3.6	192	128	3	3.3	>2500
TI	Msp430F149	2000	16	1.8-3.6	2	60	0.42	1.6	6
	Msp430F1611	2004	16	1.8-3.6	10	48	0.5	2.6	6
	Msp430F2618	2007	16	1.8-3.6	8	116	0.5	1.1	1
	Msp430F5437	2008	16	1.8-3.6	16	256	0.28	1.7	5
ZiLOG	eZ80F91	2004	16	3.0-3.6	8	256	50	50	3200

3.5 Hardware platform

Communication chip

Communication chip is an important part of wireless sensor node. In the energy consumption of a wireless sensor node, communication chip usually **consumes the most energy**. In the commonly used TelosB node, the current of CPU in the working state is only 500uA, while the current of communication chip in the working state is nearly 20mA.

Low-power communication chips consume very little energy in the sending and receiving states, which means that as long as the communication chip is on, it USES about the same amount of energy

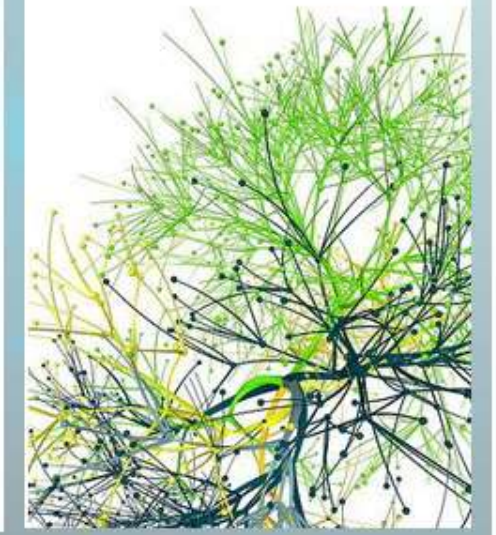


3.5 Hardware platform

Communication chip(Next)

Transmission distance of communication chip is an important index to select sensor node. The higher the transmitting power, the higher the receiving sensitivity and the farther the signal transmission distance.
Common communication chips:

- CC1000: can work at 433MHz, 868MHz and 915MHz;When using serial communication mode, the speed can only reach 19.2Kbps
- CC2420: working frequency is 2.4ghz, which is a chip completely in compliance with IEEE 802.15.4 protocol specification. Transfer rate of 250 KBPS



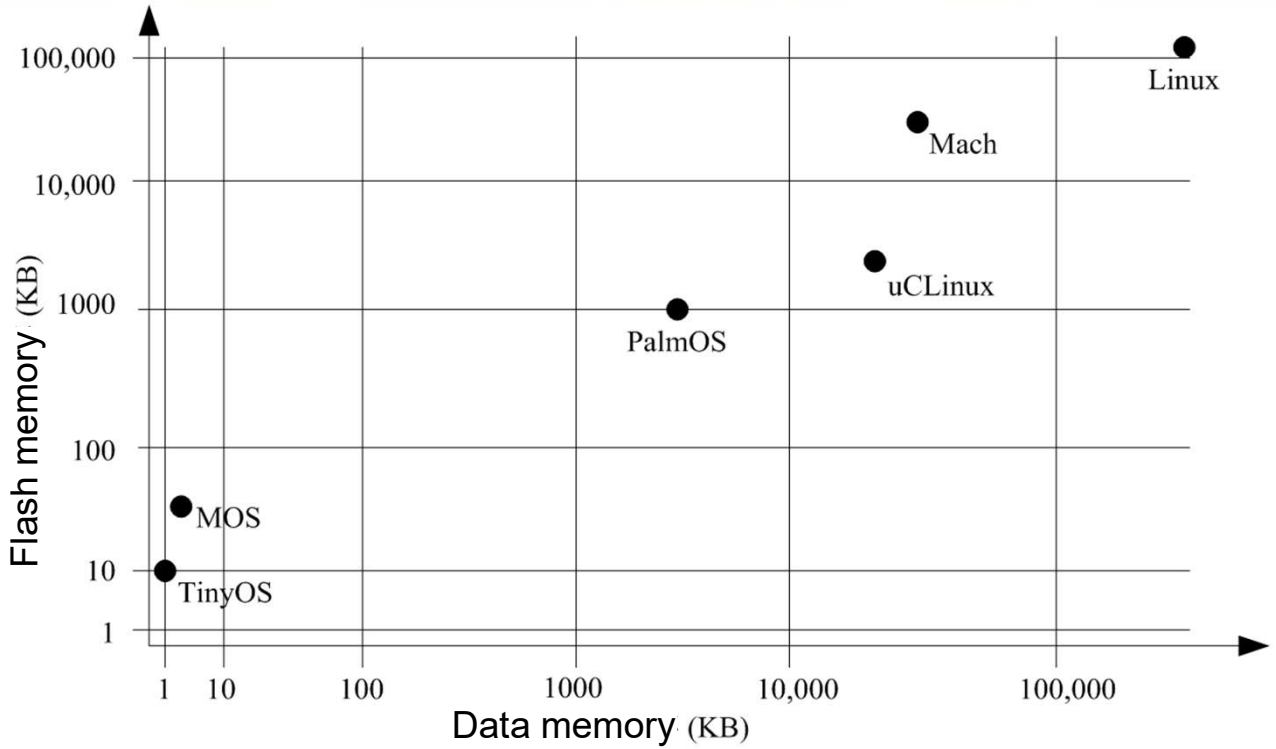

Common communication chips and their key characteristics

Manufacturer	Equipment	Year	Wake Time(ms)	Receive sensitivity(dbm)	Transmitting power(dbm)	Power consumption during receiving(mA)	Power consumption during transmitting(mA)	Power consumption during sleeping(mA)
Atmel	RF230	2006	1.1	-101	3	15.5	16.5	0.02
Ember	EM260	2006	1	-99	2.5	28	28	1
Freescale	MC13192	2004	7-20	-92	4	37	37	1
	MC13202	2007	7-20	-92	4	37	37	1
	MC13212	2005	7-20	-92	3	37	37	1
Jennie	JN5121	2005	>2.5	-93	1	38	38	<5
	JN5139	2007	>2.5	-95.5	0.5	37	37	2.8
TI	CC2420	2003	0.58	-95	0	18.8	18.8	1
	CC2430	2005	0.65	-92	0	17.2	17.2	0.5
	CC2520	2008	0.50	-98	5	18.5	18.5	0.03

Content

The operating system is the core of sensor node software system.

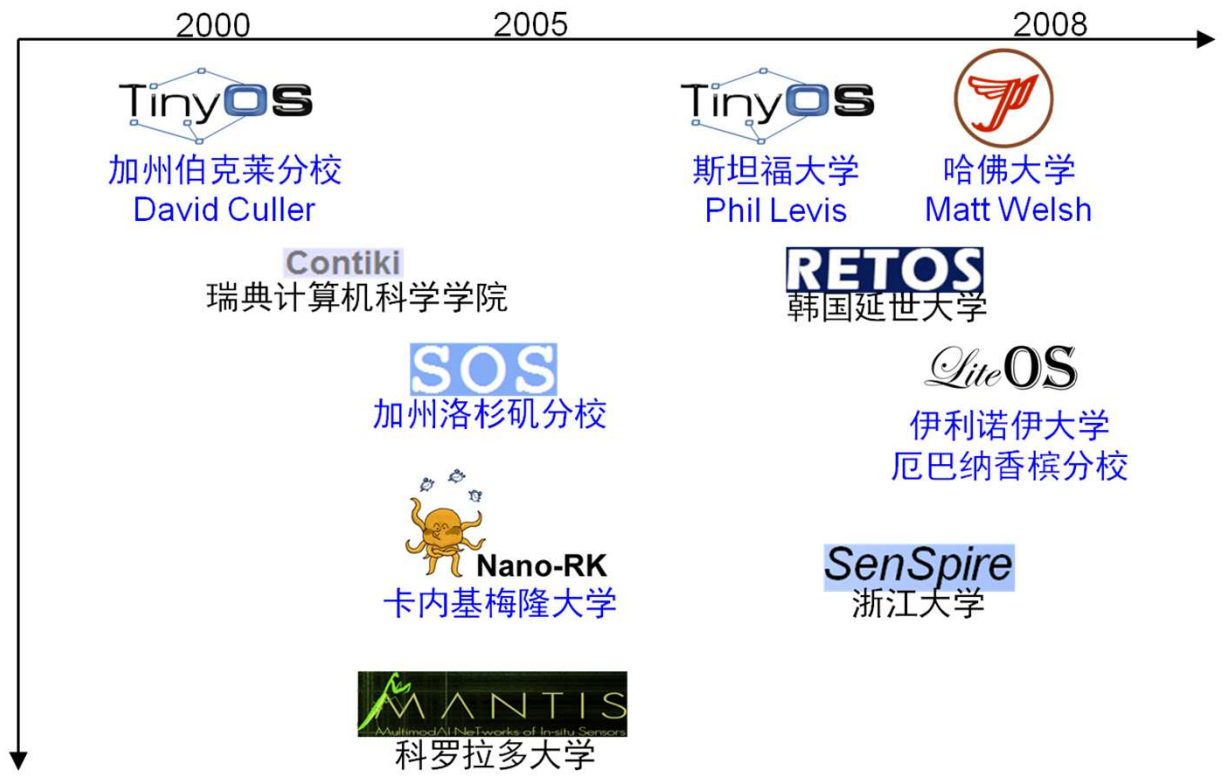
Node operating system vs. Other operating systems



As can be seen from the figure, the node operating system is extremely miniaturized.

Node operating systems are different from traditional operating systems in that **their hardware platform resources are extremely limited**

History of node operating system



Node operating systems are different from traditional operating systems in that **their hardware platform resources are extremely limited**

TinyOS

Developed by UC Berkeley, TinyOS is the most widely used OS in wireless sensor network research (http://www.tinyos.net).

TinyOS development language: nesC

- nesC is a development language designed specifically for sensor nodes with limited resources and diverse hardware platforms
- Applications written with nesC are component-based
- The interaction between components must be through the use of interfaces
- Applications written in nesC typically have a top-level configuration file

Node operating system functions:

- Hardware driver
- Resource management
- Task scheduling
- Programming excuse

Examples of TinyOS code

```
// BlinkC.nc
module BlinkC {
  uses interface Timer<TMilli> as Timer;
  uses interface Leds;
  uses interface Boot;
} implementation {
  event void Boot.booted() {
  call Timer.startPeriodic( 250 );
}
  event void Timer.fired() { call Leds.led0On(); }
}
```

In the left code:
A BlinkC is a component that USES three interfaces: Timer, Leds, Boot.

In the implementation section, it can invoke services provided by these interfaces, such as Timer.startPeriodic, to start a clock triggered by a 250ms cycle, and Leds.led0Toggle lights up the first light.

In the code above, notice that the event keyword represents the system event handled by the BlinkC component.


Examples of TinyOS code

```
// BlinkCApp.nc
configuration BlinkAppC {}
implementation {
  components MainC, BlinkC, LedsC;
  components new TimerMilliC() as TimerC;
  BlinkC -> MainC.Boot;
  BlinkC.Timer -> TimerC;
  BlinkC.Leds -> LedsC;
}
```

The code on the left shows a typical nesC configuration file. It must specify which components are used by the current program. For example, the program uses MainC, BlinkC (the component shown in code 1), LedsC, and TimerC components.

Which component provides the interface used in the BlinkC component? For example, the Boot interface used by the BlinkC component is provided by the MainC component. The Timer interface used by the BlinkC component is provided by the TimerC component. The Leds interface used by the BlinkC components is provided by the LedsC components.

TinyOS(Next)

TinyOS's Task scheduling

- The TinyOS core uses an event-driven single-threaded task scheduling mechanism, which is quite different from the multi-threaded scheduling mechanism of traditional OS
- The processor can only perform one task at any one time. Therefore, if a task is currently executing, the processor must wait for this task to finish before it can start processing another task
- There can be no blocking calls such as IO in a single TinyOS task

Node operating system functions:

- Hardware driver
- Resource management
- Task scheduling
- Programming excuse

Other commonly used micro OS comparisons

	TinyOS	Contilki	SOS	Mantis	NanoRk	RETOS	LiteOS
Conference	ASPLOS(2000)	EmNet(2004)	MobiSys(2005)	MONET(2005)	RTSS(2005)	IPSN(2007)	IPSN(2008)
Dynamic/Static	static	dynamic	dynamic	dynamic	static	dynamic	dynamic
Event Driven/Multithreading	Event Driven&Multithreading TOSThreads	Event Driven&Multithreading	Event Driven	Multithreading&Event Driven TinyMOS	Multithreading	Multithreading	Multithreading
Monokaryon/Modularity	Monokaryon	Modularity	Modularity	Modularity	Monokaryon	Modularity	Modularity
Network layer	Active message	uIP,uIPv6,Rime	messgae	"comm" layer	socket	three-tier architecture	
Online support	No	No	No	No	Yes	Suit POSIX 1003.1b	No
Language support	nesC	C	C	C	C	C	LiteC++

Node operating system functions:

- Hardware driver
- Resource management
- Task scheduling
- Programming excuse

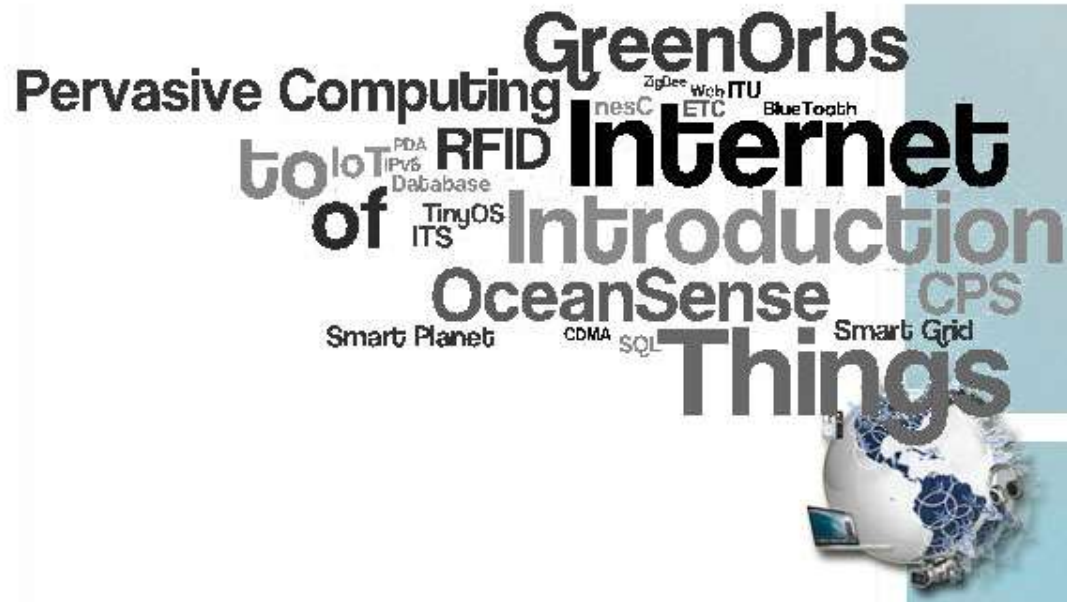
Conclusion

Review

This chapter introduces the basic concepts and typical applications of sensors, discusses the design requirements of sensors and hardware and software platforms, and takes TinyOS as an example to briefly introduce the node operating system.

Key Points

- The basic components of modern sensors and the characteristics and requirements of hardware and software platforms of each part.
- Grasp the bottleneck restricting the improvement of sensor performance and corresponding design requirements (low cost and miniaturization, low power consumption, flexibility and scalability, robustness)
- Understand the main features of the node operating system and the basic framework of TinyOS/nesC programming



Thank you!

Internet of Things