Development of an energy efficient, robust and modular multicore wireless sensor network

Hong-Ling Shi

N° d'ordre : D. U. 2435

EDSPIC : 641

UNIVERSITÉ BLAISE PASCAL - CLERMONT II

ÉCOLE DOCTORALE DE

SCIENCES POUR L'INGÉNIEUR DE CLERMONT-FERRAND

Thèse

Présentée par

Hong-Ling SHI

Pour obtenir le grade de

DOCTEUR D'UNIVERSITÉ

Spécialité : INFORMATIQUE

Development of an Energy Efficient, Robust and Modular Multicore Wireless Sensor Network

---

Soutenue publiquement le 23 janvier 2014 devant le jury :

Directeur de la thèse :
Prof. Kun Mean HOU

Membres du jury :
Prof. Bernard Tourancheau (University of Joseph Fourier, rapporteur)
Prof. Houda Labiod (ENST ParisTech, rapporteur)
Prof. Alain Quilliot (UBP, examinateur)
Dr. Jean-Pierre Chanet (IRSTEA, examinateur)
Prof. Jorge Garcia-Vidal (UPC, examinateur)
Dr. Jianjin LI (invité)
Dr. Christophe de Vaulx (invité)

Remerciements

First and foremost, I would like to thank my supervisor, Prof. Kun Mean HOU, who gave me an opportunity to do this challenging and interesting research. He is a constant source of inspiration and always helpful when I need help. I learned so such from him and I believe these unforgettable experiences and valuable knowledge will be a great asset in my future careers as well as in my personal life.

I would like to thank Conseil Régional d'Auvergne and Feder for their financial support. I would also like to thank Laboratory LIMOS- UMR 6158 - Universit é Blaise Pascal/CNRS for providing all necessary equipment to do the research.

I am very grateful to all the members of the jury for their valuable time and feedback. I am very thankful to Prof. Hong SUN from Wuhan University (China) for her recommendation and encouragement. I am very grateful to Dr. Jian-Jin LI and Dr. Christophe DE VAULX for their kind help, patience and support. I am thankful to Prof. Haiying ZHOU and Prof. XiaoZhong YANG from Harbin Institute of Technology (China) for the in-depth discussions with them.

I would like to express many thanks to present and former team members of SMIR group for their efficient cooperation and kind help. Thanks to XunXing DIAO, Hao DING, Xing LIU, YiBo CHEN, Bin TIAN, Peng ZHOU, Khalid el GHOLAMI, Muhammad YUSRO, Lizhong ZHANG, Zuoqin HU, Messaoud KARA, XinChen ZHANG and Jing WU. They have all contributed to create a great work environment.

Finally, I would like to dedicate this dissertation, in loving memory, to my father. May his soul rest in peace! I would also like to dedicate this dissertation to my mother for her support, encouragement, and love over the years. I would also like to thank my sister, my brother, and all my friends who have supported me. Last, but not least, my deepest gratitude is towards my wife, Li-Li HUANG, for her love, patience, and support.

Table of Contents

List of Figures

List of Tables

List of Acronyms

6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
BI	Business Intelligence
CoAP	Constrained Application Protocol
COTS	Commercial off-the-Shelf
CRM	Customer Resource Management
DMRTT	Design for Multicore Run Time Testability
ECAM	Efficient Context Aware Middleware
EPER	Enhance PandaBoard Edge Router
ERP	Enterprise Resource Planning
E^2MWSN	Energy Efficient and Fault Tolerant Multicore Wireless Sensor Network
FIT	Failures in Time Failure Rate in Parts per Billion Hours
HEROS	Hybrid Event-driven and Real-time multitasking Operating System
HRDP	High Reliability Design Process dedicated to High Resource Constraint Embedded System
HSDTVI	Hardware Support Debug Test and Validation Interface
IEC	International Electrotechnical Commission
iLive	First generation Intelligent Limos Versatile Embedded WSN
iLiveEdge	Intelligent LImos VersatileEmbedded Edge Router
IWoT	Internet of Things &Web of Things
IoT	Internet of Things
LiveNode	Limos Versatile Embedded wireless sensor NODE
MTBCF	Mean Time between Critical Failure
MTBF	Mean Time between Failure
NC	Nano Controller
PGIS	Parking Guidance and Information System
PSU	Power Supply Unit
RAS	Reliability, Availability, and Serviceability
RTC	Real-time Clock
SIS	Smart Irrigation System
SMIR	Systèmes Multisensoriels Intelligents intégrés et Répartis
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver Transmitter
USART	Universal Synchronous/Asynchronous Receiver Transmitter
VLSI	Very Large Scale Integration
WN	Wireless Sensor Node
WoT	Web of Things
WSN	Wireless Sensor Network

Chapter 1. Introduction

1.1. Motivation and Problem Statement

Wireless Sensor Network (WSN) has attracted more and more attentions from both scientific community and governments around the world in recent years. It is considered as a key technology of the 21st century and as the foundation of Pervasive computing, Mobile computing, Wearable computing (Body Area Network 'BAN' etc.) and Internet of Things (IoT). WSN is composed of a set of WSN nodes (Wireless Sensor Network nodes) equipped with different types of sensors and linked with each other by wireless access medium. These WSN nodes can collaborate to perform distributed sensing tasks. The advances in Very Large Scale Integration (VLSI) chip designs, wireless network and Micro-Electro-Mechanical Systems (MEMS) enable the development of smarter, smaller, cheaper and low energy consumption WSN nodes powered by battery.

The WSN nodes use battery as power supply source; connect each other through wireless medium and form a network through self-organization methods (RPL). Therefore, WSN can work without preinstalled wire or existent infrastructure. This key feature enable WSN to be easily and cheaply deployed in areas of interest, which normally very difficult or impossible to access such as inaccessible terrains, moving people and animal, disaster places, and so on. The WSN can serve as an interface to the real world and fulfill the gap between real world and information systems. The smart tiny nodes can sense the environment through different type of sensors; gather information such as temperature, humility, distance, speed, pressure, light, pollution, etc.; make local decisions based on the related information; transfer user interested information to the center server; interact remotely with user through wireless link. Some of them even have the capability to act on the environment through actuators such as electro-valve, alarm speaker etc. These special WSN nodes make a special type of WSN: Wireless Sensor and Actuator Network (WSAN). In this dissertation, WSAN will be considered as a subset of WSN without specific distinction.

The wide varieties of user interest to real world make wide range of WSN applications. J. Yick classifies them into two categories: monitoring and tracking in (Yick, Mukherjee, & Ghosal, 2008). (see Figure 1-1). Monitoring applications include indoor/outdoor environmental monitoring, health and wellness monitoring, power monitoring, inventory location monitoring, factory and process automation, and seismic and structural monitoring. Tracking applications include tracking objects, animals, humans, and vehicles (Yick et al., 2008).

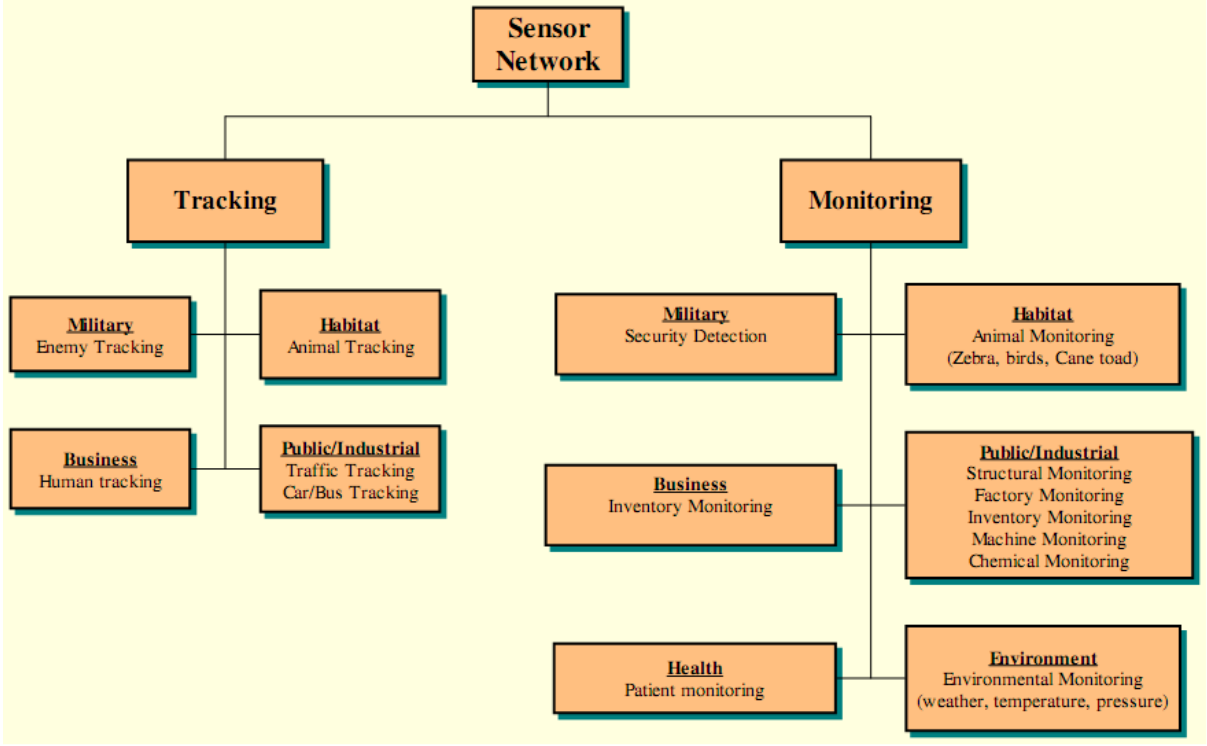

Figure 1-1 Overview of WSN applications(Yick et al., 2008)

Nowadays there are many WSN nodes such as BTnodes, l'ESB/2 nodes, SmartTags, EYES node, TinyNode, Mote, Mica2, Tmote Sky, Atlas and Imote (Akyildiz, Su, Sankarasubramaniam, & Cayirci, 2002; Baronti et al., 2007; Basaran et al., 2006; Yick et al., 2008) implemented to fulfill the huge amount of applications. Note that, all these WSN nodes are quite similar in term of functionality. They are based on 8 or 16-bit RISC microcontroller, such as Atmel AVR (Atmel-Corporation, 2012b), MSP430 (Texas-Instruments, 2012) etc., equipped with a unique Bluetooth or ZigBee wireless access medium having 200m LOS 'Line Of Sight' range (Basaran et al., 2006), and enable to implement a simple wireless sensor application. These WSN nodes have been designed with highly resource constraint, so the fault tolerant is not highly considered during the design process.

Notice that in real world applications, WSN nodes will be deployed in open harsh environment. They may suffer different faults such as physical damage, environmental hazards, interference etc. Therefore based on these faulty sensor nodes, researchers have developed many techniques to increase the reliability of the whole network. Redundant use of WSN nodes, reorganization of sensor network, and overlapped sensing regions are few of the techniques employed to increase the fault tolerance or reliability of the network (Gao, Xu, & Li, 2007; Hsieh, Leu, & Shih, 2010; Khan, Daachi, & Djouani, 2012; M.-H. Lee & Choi, 2008; Nakayama, Ansari, Jamalipour, & Kato, 2007).

This dissertation focuses on developing a reliable WSN from the early stage. We want to develop a more reliable WSN node comparing with the existing one. To achieve this goal, we will introduce a new design process to develop and implement WSN node. The reliability of Very Large Scale Integration (VLSI) is highly improved during recent decades. One of the

most important reasons for the success of VLSI technique is that the whole industry makes Design for Testability (DFT) as their industry standard. DFT makes it possible to assure the detection of all faults in a circuit, reduce the cost and time associated with test development, and reduce the execution time of performing test on fabricated chips.

Figure 1-2 shows the block diagram of a simple DFT system.

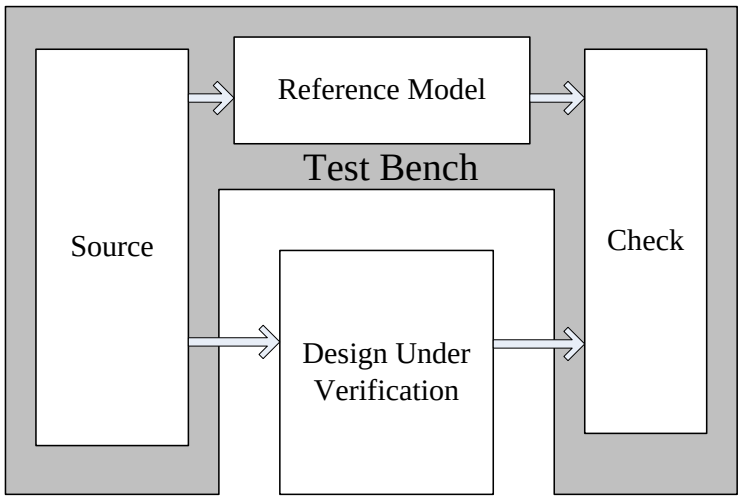

Figure 1-2 Block diagram of DFT system

As we expect our WSN node to be more reliable, and DFT exactly meets this requirement. Therefore, our motivation is applied the DFT concept to our WSN design process in order to improve its reliability.

1.2. Our Multicore Solutions

Figure 1-3 presents the block diagram of a multicore DFT solution for WSN. To transplant the DFT concept to a new WSN node, first we put the center microcontroller, application core, of tradition sensor node as Design under Verification (DUV) module. Then, we add another core to run the Test Bench (TB) module. We name this core as Fault Detect and Fault Recover Core (FD & FR Core). We use separate core to test and validate the application core in order to avoid the intra-system interference.

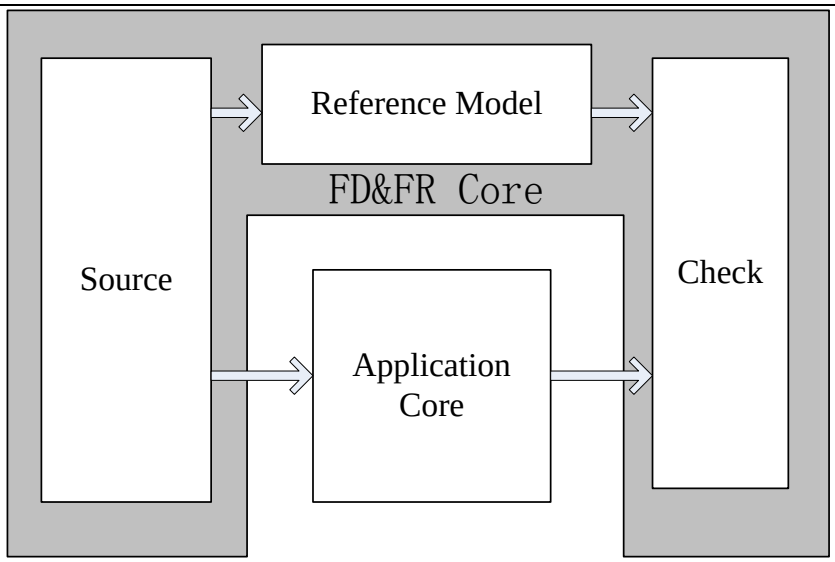

Figure 1-3 Block diagram of Multicore WSN node

When faults show up in the application core, which is running the user application software, the FD & FR Core can detect and identify the faults, similarly as TB module do in module/chip validation process, then help the faulty node recover from these faults. The multicore WSN node can tolerate these faults automatically. Because the test and validation process in multicore WSN node is carried out in real-time, so we name this method as Design for Multicore Run Time Testability (DMRTT). Moreover, the application core in DUV module may have a redundancy for further improving reliability as shown in Figure 1-4.

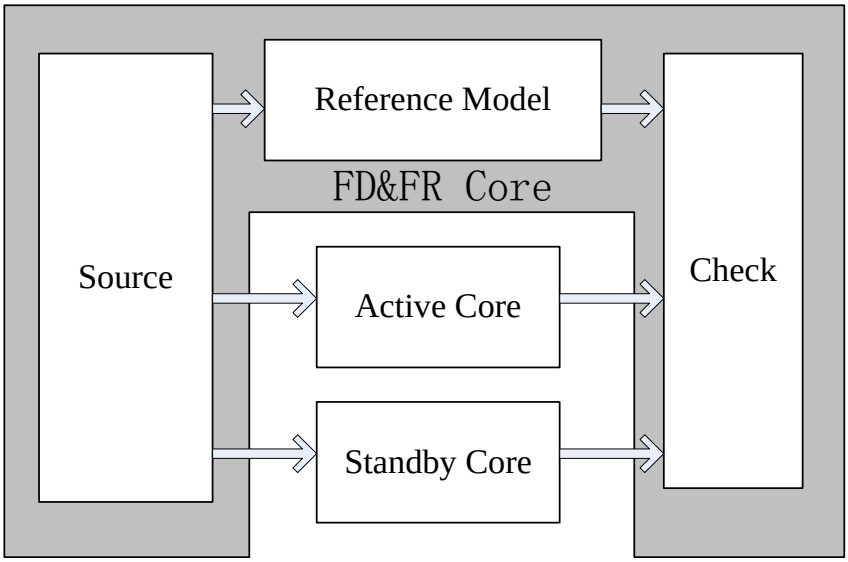

Figure 1-4 Block diagram of Multicore WSN node with redundancy for application core

It is known to all that the energy constrain of WSN node is very high. Normal microcontroller is too big in form factor and high in power consumption to be the FD & FR Core. Therefore, we introduce Nano Controller (NC) to be the FD & FR Core. NC is a very small and ultra-low power consumption controller. Because it is very small, so it will not notably affect the cost and complexity of WSN node. Moreover, it is an ultra-low power

consumption controller, which consumes only one percent energy of normal microcontroller, so it can help WSN node to be more energy efficient when the application core is switched off. DMRTT and NC together form our multicore architecture, which can greatly improve the reliability and energy efficiency of WSN node without significant increase in cost and complexity.

1.3. Contributions

The contributions of this dissertation are in the area of fault tolerance, specifically in the field of system design and hardware platform design of fault tolerance WSN. Overall, the main contributions of this dissertation are:

- We first present a novel modular multicore architecture that meets the strict dependability and energy efficiency requirements of wireless sensor networks. The multicore architecture can highly improve the reliability of WSN without sacrificing simplicity.
- Then, we present a design process (High Reliability Design Process dedicated to Resource Constraint Embedded System: HRDP) based on multicore architecture to ease the development of dependable WSN. The HRDP can easily adjust to apply in any resource constrained embedded system to improve the reliability.
- Finally, we demonstrate the flexibility of our multicore architecture on several hardware platforms, E²MWSN, iLive, SIS, iLiveEdge, EPER, RPiER, etc. These hardware platforms already form some WSN in different long-term, battery operated real-world applications. They can meet the application requirements very well.

1.4. Dissertation Structure

This dissertation has seven chapters. The remainder of the dissertation is organized as follows:

Chapter 2, General Purpose Dependable System, presents a survey of dependable system. It tries to provide the necessary background for a general understanding of the issue discussed in later chapters.

Chapter 3, Dependable Wireless Sensor Networks, provides a general overview of the dependable WSN. By considering the needs of applications, this chapter describes the shortcomings of traditional wireless architectures and current approaches motivates design choices made later in this dissertation.

Chapter 4, Multicore WSN Node Architecture, presents the overall framework, multicore architecture, to address the dependability of WSN. The designing goal of our multicore

architecture is trying our best to improve the reliability of WSN without significant increasing cost and complexity.

Chapter 5, High Reliability Design Process dedicated to Resource Constraint Embedded System, discusses the design process (High Reliability Design Process dedicated to Resource Constraint Embedded System: HRDP) that developers may carry out to make full use of multicore architecture. The HRDP and multicore architecture are both technologies independent, they both can easily adjust to any resource constrained embedded system to improve the reliability.

In Chapter 6, Implementation of Multicore Wireless Sensor Node, we present several hardware platforms, E³MWSN, iLive, SIS, iLiveEdge, EPER, RPiER, etc., and several real-world applications based on these platforms. These hardware platforms will demonstrate the flexibility of our multicore architecture. Additionally the applications will validate the real-world performances of our architecture.

Chapter 7 summarizes the thesis and concludes with a prediction of future technological trends.

Chapter 2. General Purpose Dependable System

The dependability of computer system has been a challenge ever since computers first appeared in the middle of the 20th century. In those days, computers were built by using unreliable components such as vacuum tubes, relays, and so on. They were expensive, and used mainly by government and big corporations.

Nowadays, computers are built from more reliable components, such as semiconductor components, and other components from more advanced technology. With the ever-increasing circuit density, computers are more reliable and no longer expensive commodities thanks to fault detections and corrections techniques (Blundell, 2007). They becoming an every-day commodity, deeply embedded in practically every aspect of our lives, from visible desktops, laptops, smart phones etc., to invisible vital components of cars, home appliances, medical equipment, aircraft, industrial plants, and power generation and distribution systems.

As we are increasingly dependent on services provided by computer systems and our vulnerability to computer failures is also growing. We would like these systems to be dependable: they should still deliver an acceptable level of service in spite of faults. Notice that how to design a low cost robust embedded system is still a challenge.

In this chapter, we will present a survey of the techniques dedicated to dependable system. These techniques will be the fundamentals of our target fault tolerant wireless senor network.

2.1. Introduction

Dependability is defined in (A. Avizienis, Laprie, Randell, & Landwehr, 2004) as the ability to deliver service that can justifiably be trusted. It also encompasses mechanisms designed to increase and maintain the dependability of a system. Dependability covers the availability performance and its influencing factors: reliability performance, maintainability performance and maintenance support performance (including management of obsolescence). The International Electrotechnical Commission (IEC), via its Technical Committee TC 56 develops and maintains international standards in the field of dependability (IEC, 2013).

Before giving more details on different technical methods for improving dependability, we firstly discuss overview of dependability concepts.

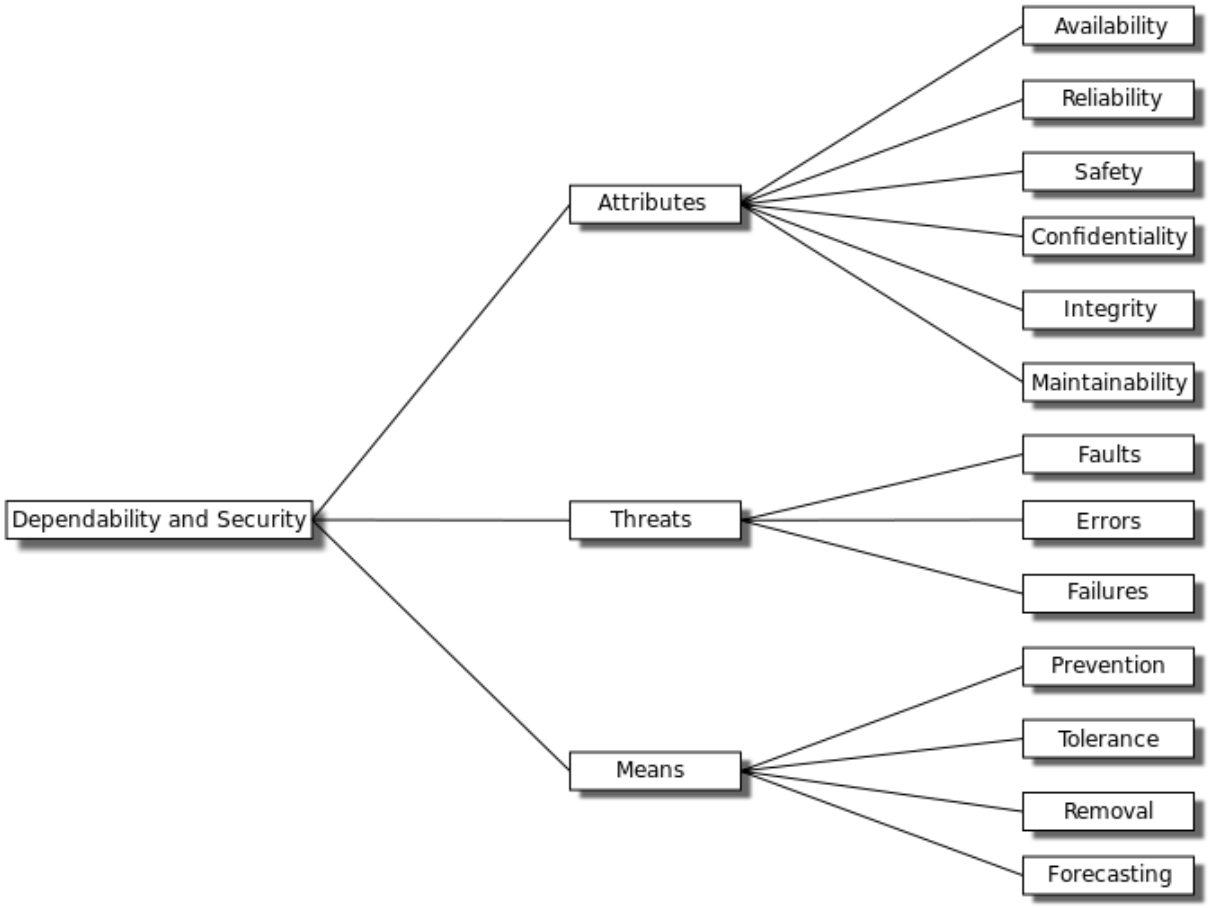

Figure 2-1 The dependability tree (Algirdas Avizienis, Laprie, & Randell, 2001)

Figure 2-1 shows a systematic exposition of the concepts of dependability (Algirdas Avizienis et al., 2001), it can be broken down into three elements:

Attributes - A way to assess the dependability of a system;

Threats - An understanding of the things that can affect the dependability of a system;

Means - Ways to increase a system's dependability;

Dependability is a generic concept that is led by three groups of fundamental concepts: its attributes, the threats to its attainment and the means to reach the desired dependability goals.

2.1.1. Dependability Attributes

The dependability attributes represent different aspects of the service delivery. They are used to express and analyze the quality of the service delivered or expected from the system. Based on the needs of the user(s), several kinds of attributes can be found, but they are almost compositions or specializations of the following basic ones:

- Availability (A(t)): The probability that a system is operating correctly and is available to perform its functions at the instant of time t.

- Reliability (R(t)): The conditional probability that a system has functioned correctly throughout an interval of time, [t_0, t], given that the system was performing correctly at time t_0.
- Safety (S(t)): The probability that a system will either perform its functions correctly or will discontinue its functions in a well-defined, safe manner.
- Confidentiality: absence of unauthorized disclosure of information
- Integrity: absence of improper system state alterations
- Maintainability (M(t)): The probability that an inoperable system will be restored to an operational state within the time t.

2.1.2. Dependability Threats

The threats to dependability are faults, errors and failures. They are the circumstances at the origin of an incorrect service delivery. Their effects deteriorate the level of satisfaction of the dependability attributes.

- Fault: A physical defect, imperfection, or flaw that occurs in hardware or software; A fault is the adjudged or hypothesized cause of an error. A fault is active when it produces an error, otherwise it is dormant
- Error: The occurrence of an incorrect value in some unit of information within a system; An error is that part of the system state that may cause a subsequent failure
- Failure: a deviation in the expected performance of a system; A failure occurs when an error reaches the service interface and alters the service, i.e., system cannot provide correct system function.

2.1.3. Dependability Means

The development of a dependable computing system calls for the combined utilization of a set of four techniques:

- Fault prevention: A technique that an attempts to prevent the occurrence of faults; It is more related to general engineering processes and is handled by quality control techniques employed during design and development of systems.
- Fault tolerance: The ability to continue the correct performance of functions in the present of faults; It is carried out via the implementation of error detection and system recovery mechanisms.

- Fault removal: A technique that deals with how to reduce the number or severity of faults; It can be carried out both during the development phase, and during the use phase of a system. In development phase, it consists of verification, diagnosis and correction. In use phase, it consists in a corrective or a preventive maintenance.
- Fault forecasting: A technique that deals with how to estimate the present number, the future incidence, and the likely consequences of faults. It is conducted by carrying out an evaluation of the system behavior with respect to fault occurrence or activation.

In order to improve the dependability, the combinations of those techniques are strongly recommended. In this dissertation, the architecture of our WSN node is directly support fault tolerance; we will use fault removal in the development phase to help verify each components and whole system; fault prevention is handled thought out all the design process.

2.2. Dependability Evaluation Techniques

As Peter Drucker once said: "If you can't measure it, you can't manage it." (Brown, 1982) If we cannot estimate the dependability of present and the future candidate design, we cannot make good decisions. Therefore, we need use dependability evaluation techniques in design process to help to estimate the dependability of design, and then improve it.

2.2.1. Dependability Terms

2.2.1.1. Failure rate

A failure rate λ is the expected number of failures per unit time. For example, if a processor fails, on average, once every 1000 hours, then it has a failure rate λ = 1/1000 failures/hour. (Dovich, 1990) (IEC, 2013)

The failure of a system is

$$\lambda = \sum_{i=1}^{N} \lambda_i \tag{2.1}$$

While λ_i is the failure rate of sub system.

From failure rate λ, we can have Reliability (R(t)):

$$R(t) = e^{-\lambda t} \tag{2.2}$$

The common used unit of failure rate is FIT (Failures in Time Failure Rate in Parts per Billion Hours). One FIT equals one failure per billion (10^9) hours (once in about 114,155 years). The FIT is especially good for the failure rate of individual components, since their failure rates are often very low.

Figure 2-2 and Figure 2-3 provide a typical evolution of failure rate over a lifetime of a hardware system and a software system (Dubrova, 2013).

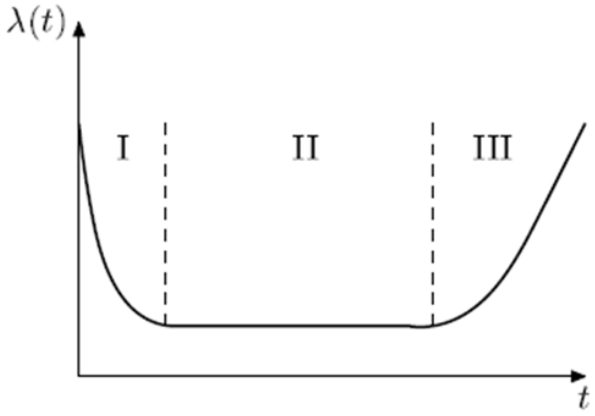

Figure 2-2 Typical evolution of failure rate over a lifetime of a hardware system (Dubrova, 2013)

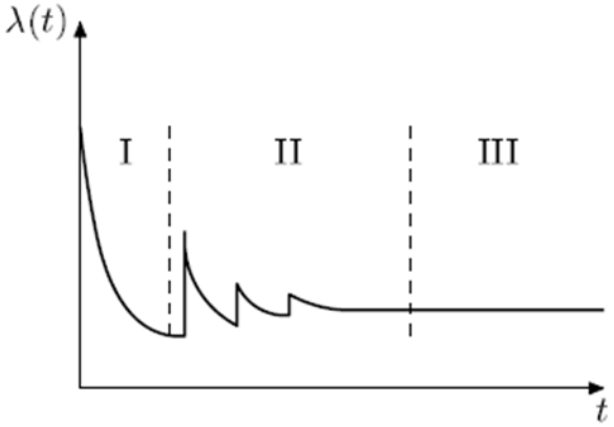

Figure 2-3 Typical evolution of failure rate over a lifetime of a software system (Dubrova, 2013)

As shown in Figure 2-2, hardware failures rate can typically characterize by a bathtub curve. The chance of a hardware failure is high during the initial life of the module (Phase I). The failure rate during the rated useful life (Phase II) of the product is low. Once the end of the life (Phase III) is reached, failure rate of modules increases again.

Software failures rate, however, does not show the same characteristics similar as hardware. A possible curve is shown in Figure 2-3 if we projected software failures rate on the same axes (Reliability Analysis Center, 1996). There are two major differences between hardware and software curves. One difference is that in the last phase (Phase III), software does not have an increasing failure rate as hardware does. In this phase, software is approaching obsolescence; there is no motivation for any upgrades or changes to the software. Therefore, the failure rate will not change. The second difference is that in the useful-life phase (Phase II), software will experience a drastic increase in failure rate each time an upgrade is made. The failure rate levels off gradually, partly because of the defects found and fixed after the upgrades.

2.2.1.2. Mean time to failure

Another important and frequently used measure of interest is mean time to failure defined as follows. The *mean time to failure* (MTTF) of a system is the expected time of the occurrence of the first system failure (Dubrova, 2013).

$$MTTF = \int_0^\infty R(t)dt \tag{2.3}$$

$$MTTF = \int_0^\infty e^{-\lambda t}dt = [\frac{1}{-\lambda}e^{-\lambda t}]_0^\infty = \frac{1}{\lambda} \tag{2.4}$$

2.2.1.3. Mean time to repair

The *mean time to repair* (MTTR) of a system is the average time required to repair the system. MTTR is commonly specified in terms of the *repair rate* μ, which is the expected number of repairs per unit time (Dubrova, 2013):

$$MTTR = \frac{1}{\mu} \tag{2.5}$$

From the definition of MTTF and MTTR, we can have Availability:

$$Availability = \frac{MTTF}{MTTF + MTTR} \times 100\% \tag{2.6}$$

2.2.1.4. Mean time between failures

The *mean time between failures* (MTBF) of a system is the average time between failures of the system. The MTBF should be used as part of a model that assumes the failed system will be repaired immediately (zero elapsed time) as opposed to mean time to failure (MTTF), which measures average time between failures of non-repairable systems only. However, in practice, MTBF is commonly used for both types of systems, repairable and non-repairable (Reliability Information Analysis Center, 2005; Zzyzx Peripherals, 2001).

MTBF is a measure of how reliable a hardware product or component is. It describes the flat, bottom of the bathtub curve of failure rate. MTBF is equal to the inverse of failure rate.

$$MTBF = \frac{1}{\lambda} \tag{2.7}$$

Note that many products with very low failure rates during "normal life" will wear out in a few years, so that the Lifetime may be much less than MTBF. The Figure 2-4 shows the relationship between MTBF and Lifetime.

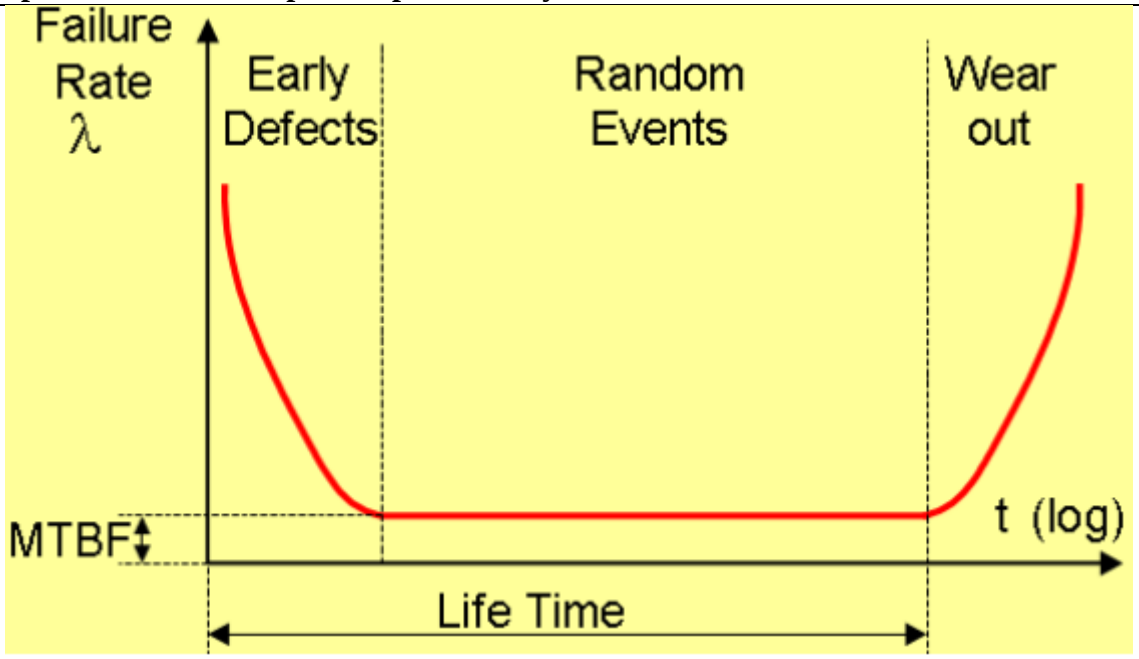

Figure 2-4 MTBF versus Lifetime

Unlike the hours from the MTBF calculations, lifetime indicates the operating hours expected under normal operating conditions. The lifetime is the period of time between starting to use the device and the beginning of the wear-out phase. This period of time is determined by the life expectancy of the components used in the assembly of the unit. As with any design, the weakest component with the shortest life expectancy determines what the life of the whole product will be. For example in power supplies, the electrolytic capacitors have the shortest lifetime expectancy.

So the lifetime is determined by the weakest component, so the redundancy cannot help to improve the lifetime. Meanwhile the MTBF can be highly affected by the architecture (redundancy e.g.). In case of Dual Modular Redundant (DMR) system with same component, each component has MTBF1000 hours, the MTBF of the DMR system is 1,000,000 hours. While in case of Triple Modular Redundant (TMR) system with same each component has MTBF1000 hours, the MTBF of the TMR system will increase to 1,000,000,000 hours.

2.2.1.5. Mean time between critical failures

The *mean time between critical failures* (MTBCF) of a system is the average time between critical failures of the system. MTBCF is a subset of MTBF because it only counts those failures that result in a mission abort or mission failure. The reliability analyst needs to be able to distinguish between those failures that are critical to the mission versus those that are not (failures that are not critical to the mission will still need to be fixed and counted as part of the MTBF calculation)(Reliability Information Analysis Center, 2005).

2.2.1.6. Summary

In this dissertation, we mainly use MTBF and MTBCF to evaluation the dependability of different WSN architecture. Therefore, we focus on the core and necessary external components, the rest components such as PCB board, connectors, sensors, batteries, RF antenna, etc. are not taken into account.

2.2.2. Dependability model types

There are mainly two common dependability models: reliability block diagrams and Markov processes. Reliability block diagrams belong to a class of combinatorial models, which assume that the failures of the individual components are mutually independent. Markov processes belong to a class of stochastic processes, which take the dependencies between the component failures into account, making the analysis of more complex scenarios possible.

Combinatorial reliability models include reliability block diagrams, fault trees, success trees and reliability graphs. Figure 2-5 show an example of serial and parallel two-component system.

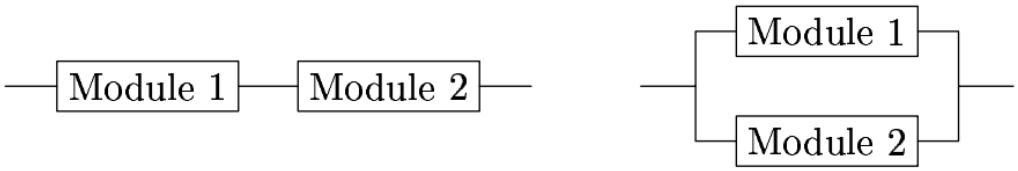

Figure 2-5 Reliability block diagram of a two-component system: (a) Serial, (b) parallel

First, reliability block diagrams assume that the system components are limited to the operational and failed states and that the system configuration does not change during the mission. Hence, they cannot model standby components, repair, as well as complex fault detection and recovery mechanisms. Second, the failures of the individual components are assumed to be independent. Therefore, the case when the sequence of component failures affects system reliability cannot be adequately represented (Reliability Analysis Center, 1996).

Contrary to combinatorial models, Markov processes take into account the interactions of component failures making the analysis of complex scenarios possible.

The WSN node is only a tiny design, so in this dissertation, we chose reliability block diagrams as the dependability models of our design.

2.2.3. Dependability computation methods

The computation methods of dependability are based on the model of dependability. Therefore, in this dissertation, we mainly discuss the computation methods based on Reliability block diagrams. Reliability block diagrams can be used to compute system reliability as well as system availability.

2.2.3.1. Reliability computation

To compute the reliability of a system represented by a reliability block diagram, we need first to break the system down into its serial and parallel parts. Next, the reliabilities of these parts are computed. Finally, the overall solution is composed from the reliabilities of the parts.

Given a system consisting of n components with $R_i(t)$ being the reliability of the ith component. If the n components are serial parts, the reliability of the overall system is given by (Dubrova, 2013)

$$R(t)_{serial} = \prod_i^n R_i(t) \tag{2.8}$$

Else, if the n components are parallel parts, the reliability of the overall system is given by (Dubrova, 2013)

$$R(t)_{parallel} = 1 - \prod_i^n (1 - R_i(t)) \tag{2.9}$$

For example, if a serial system with 100 components is to be built, and each of the components has a reliability 0.999, the overall system reliability is 0.905.

In case of parallel system having 5 components, each component has a reliability 0.96, the reliability of the system is 0.999999.

2.2.3.2. Availability computation

If we assume that the failure and repair times are independent, then we can use reliability block diagrams to compute the system availability. This situation occurs when the system has enough spare resources to repair all the failed components simultaneously. Given a system consisting of *n* components with $A_i(t)$ being the availability of the i^{th} component, the availability if the overall system is given by (Dubrova, 2013)

$$A(t)_{serial} = \prod_i^n A_i(t) \tag{2.10}$$

$$A(t)_{parallel} = 1 - \prod_i^n (1 - A_i(t)) \tag{2.11}$$

2.3. Fault Tolerance and Redundancy

As be mentioned before, in order to improve the dependability, the combinations of different dependable means should be used in the development of WSN nodes. Beyond fault removal in the development phase and fault prevention in all the design process, the ability of fault tolerance of whole design should be most important feature. It is practically impossible to foresee all the factors and run the system in a perfect environment. So the system is requested to continue the correct performance of functions in the present of faults, support fault-tolerance. Therefore, in this part, we will intro various redundancy approaches to achieve fault-tolerance.

Redundancy is the provision of functional capabilities that would be unnecessary in a fault-free environment. There are mainly two kinds of redundancy: space and time. Space redundancy provides additional components, functions, or data items that are unnecessary for a fault-free operation. Space redundancy is further classified into hardware, software and information redundancy, depending on the type of redundant resources added to the system. In time redundancy, the computation or data transmission is repeated and the result is compared to a stored copy of the previous result (Dubrova, 2013).

2.3.1. Space Redundancy

2.3.1.1. Hardware Redundancy

Hardware redundancy is achieved by providing two or more physical instances of a hardware component. For example, a system can include redundant processors, memories, buses or power supplies. Hardware redundancy is often the only available method for improving the dependability of a system, when other techniques, such as better components, design simplification, manufacturing quality control, software debugging, have been exhausted or shown to be more costly than redundancy.

There are three basic forms of hardware redundancy: passive, active and hybrid (Dubrova, 2013).

Passive redundancy achieves fault tolerance by masking the faults that occur without requiring any action on the part of the system or an operator.

Active redundancy requires a fault to be detected before it can be tolerated. After the detection of the fault, the actions of location, containment and recovery are performed to remove the faulty component from the system.

Hybrid redundancy combines passive and active approaches. It can mask the fault like in passive redundancy and reconfigure to recovery like in active redundancy. It is more reliable but more expensive than previous methods.

2.3.1.2. Software Redundancy

Reliability in software domain is still an open issue; it is not as well understood as fault-tolerance in hardware domain. There are controversial opinions on whether reliability can be used to evaluate software. Software failures are mostly due to the activation of design faults by specific input sequences. This makes the reliability of a software module dependent on the environment that generates input to the module over the time.

Many current techniques for software fault tolerance are trying to follow the same schemes of hardware redundancy. They can be divided into two groups, single-version techniques and multi-version techniques (Dubrova, 2013).

Single version techniques aim to improve fault-tolerant capabilities of a single software module. It consists of fault detection, containment and recovery mechanisms. The recovery processes use the concept of retrying the same operation in expectation that the problem is resolved after the second try.

Multi-version techniques employ redundant software modules, developed following design diversity rules. The software N-version programming closely resembles hardware N-modular redundancy.

2.3.1.3. Information Redundancy

Information redundancy techniques add extra information to date to tolerate faults. They can be divided into two types: error detecting codes and error correcting codes (Wikipedia, 2013).

Error detection is the detection of errors caused by noise or other impairments during transmission from the transmitter to the receiver. Error detecting code is most commonly realized using a suitable hash function (or checksum algorithm). A hash function adds a fixed-length tag to a message, which enables receivers to verify the delivered message by recomputing the tag and comparing it with the one provided.

Error correction is the detection of errors and reconstruction of the original, error-free data. An error-correcting code (ECC) or forward error correction (FEC) code is a system of adding redundant data, or parity data, to a message, such that it can be recovered by a receiver even when a number of errors (up to the capability of the code being used) were introduced, either during the process of transmission, or on storage.

2.3.2. Time Redundancy

Space redundancy techniques discussed so far impact physical entities like cost, weight, size, power consumption, etc. In some applications, extra time is of less importance than extra hardware, and then time redundancy will be a better solution. Time redundancy is achieved by

repeating the computation or data transmission and comparing the result to a stored copy of the previous result. If the repetition is done twice, and if the fault, which has occurred, is transient, then the stored copy will differ from the re-computed result, so the fault will be detected. If the repetition is done three or more times a fault can be corrected. Permanent faults can also be detected by repeating computation several times using different coding schemes.

Apart from detection and correction of faults, time redundancy is useful for distinguishing between transient and permanent faults. If the fault disappears after the re-computation, it is assumed to be transient. In this case, the hardware module is still usable and it would be a waste of resources to switch it off the operation. Otherwise, if the fault is permanent, the system of course should switch off or go to a safety state to avoid affect the rest parts of other system.

2.4. MTBF Values Evaluation

The MTBF value of COST component (microcontroller e.g.) is calculated from failure rate λ, which normally provided by manufacturer. The manufacturer (Atmel (Atmel-Corporation, 2012c) e.g.) provides the FIT data of component under optimal conditions and only related to hardware. According to Blue Max Technology, "Stressing a component beyond normal usage conditions may reduce the actual MTBF to a point below the 'predicted MTBF'. Generally, reliability decreases as temperature increases, so components that are operated in warm environments with poor air flow will tend to have a lower MTBF than those operated in cool environments with good air flow." According to Military & Aerospace Technology, "For every 10ºC you increase temperatures on electronics, your MTBF will be cut in half, so the hotter the electronics get, the lower the MTBF." The temperature and the humility are not only part of the parameters, which will affect the reliability of our design. The reliability of system will also be affected by other parameters such as interference, metastability, high-energy particles, software bug, misuse of the hardware and SRAM transition fault (Autran et al., 2012). That is why the real experience of error free period is always much shorter than the theoretical MTBF provided by the manufacturer.

For example, the MTBF of a standard PC is 30,000 hours or 3.4 years (Minicom Advanced Systems Ltd., 2013). The MTBF estimates for the Intel® Server System is about 50,000 hours (Intel Corporation, 2013a). The Figure 2-6 shows the MTBF Estimates of sub and total system of Intel® Server System R1208RPMSHOR.

Subassembly (Server in 40C ambient air)	Server Model S1208RPMSHOR MTBF (hours)	FIT (flrs/10^9 hrs)
S1200V3RPM board	371,523	2,692
Power Supply - 450W MiniERPS	967,300	1,034
Cooling Fan (1-fixed fans)	490,000	2,041
Cooling Fan (2-fixed fans)	77,680	12,873
Front Panel board	8,272,282	121
HS Backplane(1U,8x2.5")	3,384,479	295
Totals without motherboard =	**61,100**	**16,364**
Totals with motherboard =	**52,400**	**19,056**

Figure 2-6 MTBF Estimates for Intel® Server System R1208RPMSHOR (Intel Corporation, 2013a)

From the Figure 2-6, we can find that the most robust sub system of Intel® Server System R1208RPMSHOR is Front Panel board. Its MTBF is 8,272,282 hours, 944 years. The server board S1200V3RPM's MTBF is 371,523 hours, 42 years. As mentioned in Figure 2-4, the MTBF/MTBCF is related to the failure rate (bottom of the bathtub curve) of the system, not the product lifetime. Therefore, we cannot say that Front Panel board can work for 944 years or server board S1200V3RPM can work for 42 years. Of course, from our own experience, we can easy to find out that server (with high MTBF) is normally more reliable than the PC (with low MTBF) and demands less reboot requirements. The MTBF trend is concurrent with our user experience.

Warranty firm Square Trade has released a research paper analyzing the failure rate for 30,000 laptops comparing brands and hardware categories in 2009 (SquareTrade, 2009). The headline news of the report is that over three years, one out of three laptops will fail, and that Asus and Toshiba laptops have the lowest failure rates, while Acer, Gateway, and HP have higher than average failure rates. Additionally, two-thirds of those problems are hardware malfunctions, while the final third are classified as accidental damage. The Figure 2-7 and the Figure 2-8 show the results in this report.

Figure 2-7 Laptop three years Failure Rates (SquareTrade, 2009)

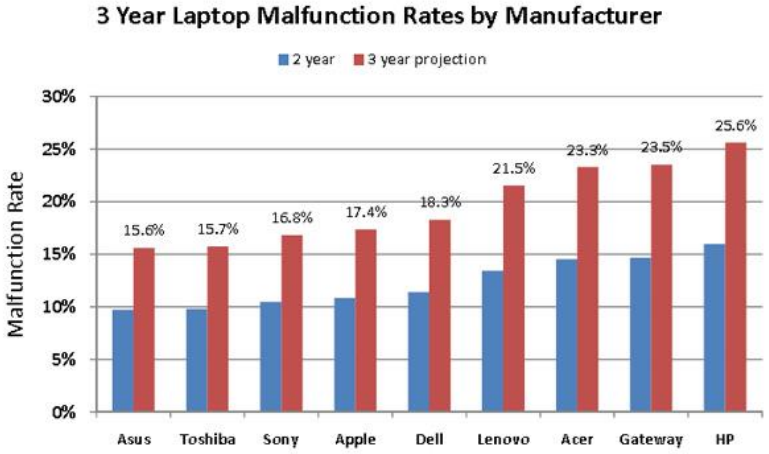

Figure 2-8 Three Years Laptop Malfunction Rates by Manufacturer (SquareTrade, 2009)

From the Figure 2-7 and the Figure 2-8, we can know that the raw MTBF is concurrent with the malfunction rates of design in product life. However, everyone has the experience to reboot PC from time to time to fix some unknown problems. Those unknown problems make the one-time useable period of PC much short than the raw MTBF. The laptops normally work indoor with user-friendly interface for user to detect the running status and manually reboot to recovery from fault. This manually fault detection and recovery mechanical enable PC resume to work until it suffer malfunction.

Meanwhile, the WSN node is working in the outdoor environment, so even its raw MTBF is relative high, and the node will not suffer malfunction in short period, but without recovery mechanical, the one-time usable period will much shorter than the raw MTBF. In our experiments, the unicore WSN network will lose 20% of its nodes in only two-week time. Therefore, we proposed to use multicore architecture to enable to implement a WSN node, which supports fault auto detection and auto recovery. By using this new multicore WSN architecture, we want to improve greatly the usable period of WSN node. Furthermore, in some of multicore WSN instances, by adopted space redundancy, we will improve both the one-time usable period and the raw MTBF of WSN node at the same time.

2.5. RAS of Computer System

Reliability, availability, and serviceability (RAS) was originally introduced by IBM as a term to describe the robustness of their mainframe computers (International Business Machines Corporation, 1970). Different operational states of a IBM server based on RAS concepts are illustrated by the Figure 2-9 (IBM Corp, 2012). This combination of hardware and software self-recovery techniques are part of advanced RAS features that increase the availability of services that must be 24x7.

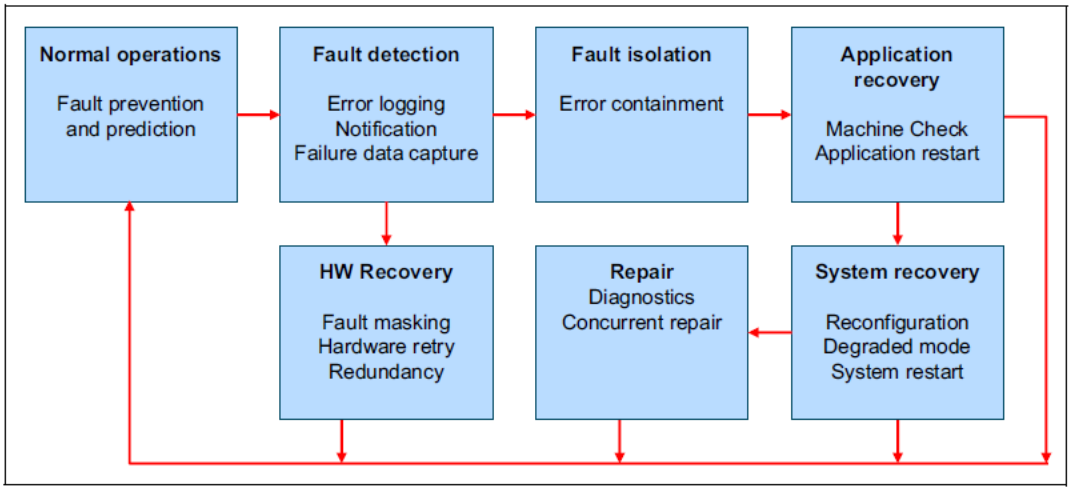

Figure 2-9 IBM server system RAS operations (IBM Corp, 2012)

For years, RAS already became a standard engineering term in computer system, especially in computer system of mission-critical applications such as database, enterprise resource planning (ERP), customer resource management (CRM), and business intelligence (BI) applications. These applications require being available 24x7 on a wide area or global basis. A failure affecting a single core business application can easily cost hundreds of thousands to millions of dollars per hour. In order to improve RAS, many approaches are adopted in different aspect of computer system. The IBM RAS server architecture is illustrated by the Figure 2-10. Moreover similar RAS system, non-stop servers, is also developed by HP (Hewlett-Packard Development Company, 2012).

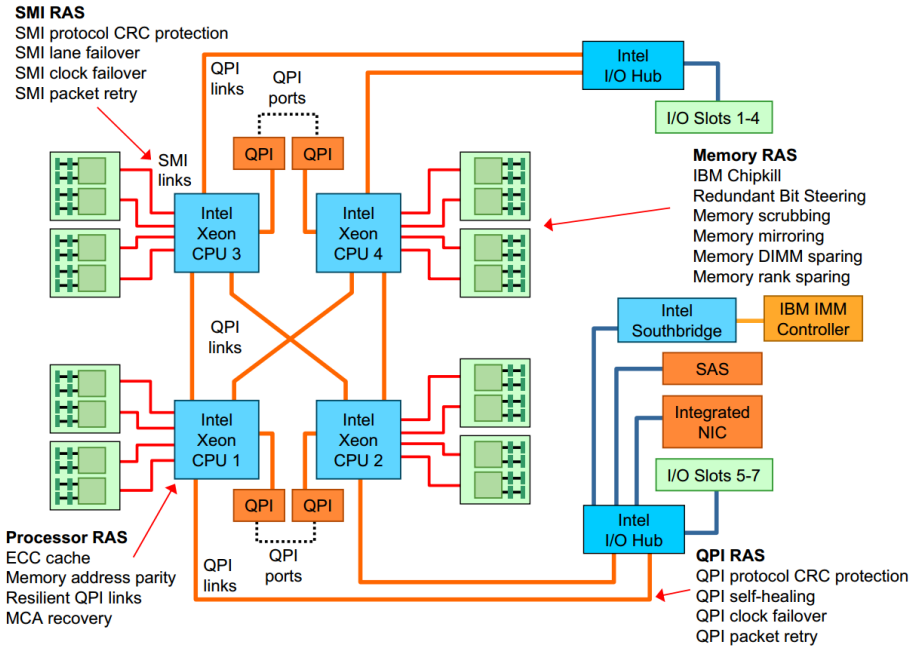

Figure 2-10 Advanced RAS features of an IBM System x3850 X5 server (IBM Corp, 2012)

The RAS architecture is symmetric space (processor, system bus, memory, devices and storage) and time redundancy (system recovery). The RAS concept used to implement very

expensive IBM and HP servers are not energy efficient. Due to the resource constraint, these approaches cannot directly use in WSN nodes. Thus, RAS concept cannot be applied to implement energy efficient multicore, modular WSN node.

2.6. Summary

In this chapter, we discussed the dependability concepts, dependability evaluation techniques, dependability terms, dependability model types, dependability computation methods, various redundancy approaches to achieve fault-tolerance, etc. Notice that, how to quantify the dependability of a real world system (HW and SW) is still an open question? The MTBF provided by the VLSI chip manufacturer is a quality indicator but it does not reflect the real world system MTBF.

In fact, a fault is a defect in hardware or software component. A manifestation of a fault, resulting in deviation from accuracy and faults may cause errors. A failure is a non-performance of expected action and errors may cause failures.

There are three types of fault: permanent, intermittent and transient. Intermittent faults occur because of unstable or marginal hardware due to environmental changes (loose connections, aging components, critical timing, interconnect coupling, resistive or capacitive variations and noise in the system). Transient faults occur because of high energy particles, temperature, humidity, pressure, voltage, power supply, vibrations, fluctuations, electromagnetic interference, ground loops, cosmic rays, alpha particles, cross talk, and electrostatic discharge. The error causes by non-permanent fault is a Soft Error 'SE'. Permanent faults reflect irreversible physical changes. The improvement of semiconductor design and manufacturing techniques has significantly decreased the rate of occurrence of permanent faults (A. C. S. Beck, Lisbôa, & Carro, 2012).

Intermittent and transient faults are expected to represent the main source of errors experiences by VLSI circuits. Failure avoidance, based on design technologies and process technologies would not fully control intermittent and transient faults.

Fault tolerant solutions, presently employed in custom design systems will become widely used in off-the-shelf ICs (Mile Stojčev 2004). For the WSN outdoor application (harsh environment), there are many soft errors. Consequently for the wide spread use of outdoor WSN robustness is a main key feature.

In order to improve the dependability of our system, the following parts of this dissertation will mainly detail on fault tolerance architecture based on active redundancy, fault removal and fault prevention is the design process.

Chapter 3. Dependability of Wireless Sensor Networks

Unlike general-purpose computing systems, WSN nodes are not easily accessible for inspection and maintenance. At the same time, the nodes have far more stringent uptime requirements than general-purpose systems; 24/7/365 uptime is usually necessary. Moreover, WSN node has high resource constraint (limited power supply, memory and CPU) and some WSN nodes are working in mission critical applications. Therefore, these real world requirements demand a great deal of research in fault tolerance and dependable wireless sensor network. The following sections will discuss different WSN nodes, WSN application, dependability threats and current approaches. The shortages of current approaches motivate the need of multicore architecture presented in this dissertation.

3.1. Wireless Sensor Networks

3.1.1. Introduction

Wireless Sensor Network 'WSN' is an active research field, which explores many technological challenges, while the WSN node design is one of the most challenging areas. The main constraints of WSN are resource and energy consumption. Consequently, the traditional embedded hardware and software solutions cannot be applied to WSN. For example, the Intel Itanium (9100 series features clock speed of up to 1.66 GHz and 667 MHz Front Side Bus (Intel Corporation, 2008)) consumes 104W. However, the energy content of a pair of alkaline AA 1.5V 2000mAh batteries is only 21.6kJ ($2 * 1.5 * 2000 * 10^{-3} * 3600 = 2.16 *10^4$ J). The power consumption of the WSN node is application dependent. It is relied on the sensor type, sample frequency, duty-cycling system, sleeping period, wireless access media, and so on. However, in order to achieve 5-years lifetime with a pair of alkaline batteries, the average power consumption must less than 137μW ($2.16 *10^4 /3600/24/365/5 = 1.37*10^{-4}$ W). Furthermore, if takes into account discharge curve of the battery voltage, the average power consumption should be even less. Thus, a WSN node should fulfill a task as a PC but consume 1 million times less energy.

A wireless sensor network is composed of a set of WSN nodes deployed in a field of interest to monitor specific phenomena. The WSN nodes can be equipped with a variety of sensors, such as air temperature sensor, air humility sensor, light sensor, soil temperature

sensor and soil moisture sensor. These WSN nodes sense specific environment phenomena, perform simple signal processing, and then send data to a central server through sink node.

WSNs can be used for a wide variety of applications dealing with monitoring (precision agriculture, environment data collection, etc.), control (disturbed sensing and controlling), and surveillance (smart care, battle-fields surveillance, etc.).

The following part will briefly introduce currently existed WSN nodes and WSN Applications.

3.1.2. WSN Nodes

Recent advances in Very Large Scale Integration (VLSI) chip designs, wireless network and Micro-Electro-Mechanical Systems (MEMS) have led to the development of low-cost, low-power, and small size WSN nodes. Different institutes or companies have developed various kinds of WSN nodes. An exhaustive survey on WSN hardware has been done by Tatiana Bokareva (Bokareva, 2013). The information on various sensors, WSN nodes, processor, radio chipsets, sensor network operating system, protocols is available at Sensor Network Museum (TIK WSN Research Group, 2013). Here we briefly introduce two types of WSN nodes: Scalar WSN nodes and Multimedia WSN nodes.

3.1.2.1. Scalar WSN Nodes

The common available Scalar WSN nodes include:

- **MICAz**

The processor board of MICAz is MPR2400, which is based on Atmel ATmega128L. The MICAz (MPR2400) IEEE 802.15.4 radio (ZigBee compliant) offers both high speed (250 kbps) and hardware network security (AES-128). Direct sequence spread spectrum radio provides resistance to RF interference and data security. The 51-pin expansion connector supports Analog Inputs, Digital I/O, I €, SPI and UART interfaces. It provides 75-100 meter of outdoor range line of sight communication (1/2 wave dipole antenna).

Figure 3-1 Circuit Board of MICAz

- **MICA2**

The processor and radio board used in MICA2 is MPR400, which is based on Atmel ATmega128L. The radio uses 868/916 MHz frequency band and supports data rate of 38.4kbps. A variety of sensors and data acquisition boards for the MICA2 mote is available which can be connected to the standard 51 pins expansion connector. Apart from its basic function as WSN node, it can also function as a base station when interfaced with MIB 510/MIB 520. The MIB510/MIB520 provides a serial/USB interface for both programming and data communications. Theoretically, it supports 150 meter of outdoor range for line of sight communication (1/4 wave dipole antenna).

Figure 3-2 Circuit Board of MICA2

- **Telos B**

The MICA2 and MICAz motes are found to be more suitable for field deployment purposes. The Telos B motes have programming and data collection facility via USB and is thus suitable for testbed deployment in lab for experimentation. It utilizes IEEE 802.15.4/ZigBee compliant radio (2.4-2.4835 GHz) which enables 250kbps of data transfer. The Telos B is based on 8 MHz TI MSP430 microcontroller with 10kB RAM. It has 1MB external flash for data logging, integrated onboard antenna and optional sensor suite including integrated light, temperature and humidity sensor.

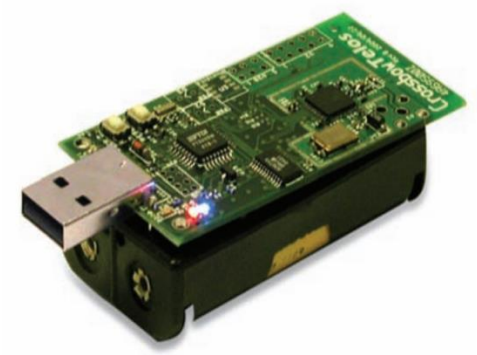

Figure 3-3 Circuit Board of Telos B

- **IRIS**

It has improved radio range as compared to MICA2, MICAz and TelosB motes. It has outdoor range over 300 meters (1/4 wave line of sight dipole antenna) and indoor range of more than 50 meters (1/4 wave line of sight dipole antenna). The radio used is IEEE 802.15.4 compliant (2.4 to 2.48 GHz) which is a globally compatible ISM band which enables 250kbps of data transfer. Apart from its basic function as WSN node, it can also function as a base station when interfaced with MIB 510/MIB 520. The MIB510/MIB520 provides a serial/USB interface for both programming and data communications. It uses XM2110CA processor board that is based on the Atmel ATmega1281. A single processor board (XM2110) can be configured to run WSN application/processing and the network/radio communications stack simultaneously. As in Mica2 and MICAz, IRIS has also a 51 pins expansion connector that supports analog inputs, digital I/O, I Ȼ, SPI and UART interfaces.

Figure 3-4 Circuit Board of IRIS

- **Cricket**

The MCS410CA, Cricket Mote, is a location aware version of the MICA2. The Cricket Mote includes all of the standard MICA2 hardware and an Ultrasound transmitter and receiver. By using ultrasound transmission, mobile devises can estimate distance.

Figure 3-5 Circuit Board of Cricket

Table 3-1 provides the detail features of these common available scalar WSN nodes.

Table 3-1 Comparison of common available scalar WSN Nodes

Feature	MICAz	MICA2	TelosB	IRIS	Cricket
Microcontroller	Atmel ATmega128L	Atmel ATmega128L	TI MSP430	Atmel ATmega1281	Atmel ATmega128L
Bus Width	8	8	16	8	8
Clock Speed (MHz)	7.373	7.373	6.717	7.373	4
SRAM	4 K	4 K	10 K	8 K	4 K
SDRAM					
EEPROM	4 K	4 K	16 K	4 K	4 K
Flash	128 K	128 K	48 K	128 K	128 K
Serial Flash	512 K	512 K	1024 K	512 K	512 K
Size (mm)	58 × 32 × 7	58 × 32 × 7	65 × 31 × 6	58 × 32 × 7	58 × 32 × 7
Battery	2 × AA	2 × AA	2 × AA	2 × AA	2 × AA
External power	2.7 V–3.3 V	2.7 V–3.3 V	2.7 V–3.3 V	2.7 V–3.3 V	2.7 V–3.3 V
Power Consumption Active (mW)	24 (3 V)	24 (3 V)	10 (3 V)	24 (3 V)	24 (3 V)
Power Consumption Sleep (μW)	75	75	8	24	75
User interface	3 LEDs	3 LEDs	USB	3 LEDs	3 LEDs
Expansion connector	51-pin	51-pin	6-pin and 10-pin	51-pin	51-pin
Serial communication	UART	UART	UART	UART	UART
Other interfaces	Digital I/O, I²C, SPI	DIO, I²C, SPI	Digital I/O, I²C, SPI	Digital I/O, I²C, SPI	DIO, I²C, SPI
Transceiver chip	CC2420	CC1000	CC2420	RF230	CC1000
Frequency band (MHz) ISM band	2400–2483.5	868/916	2400–2483.5	2400–2480	868/916
Data Rate	250 Kbps	38.4 K Baud	250 Kbps	250 Kbps	38.4 K Baud
RF Transmit power (dBm)	-24 to 0	-20 to + 5	-24 to 0	+ 3	-20 to + 5
Receive (dBm) Sensitivity	-94	-98	-94	-101	-98

From Table 3-1 we can find that those scalar WSN nodes have similar structure. They all belong to single core node, which has only one microcontroller with small memory and small computation resource. From the SRAM and Flash size and the clock speed of microcontroller, we can understand more about the real meaning of high resource constrain.

3.1.2.2. Wireless Multimedia Sensor Network ‘WMSN’ Node

In fact, the requirements of diverse environmental data collection applications (precision agriculture e.g.) become more complex. The scalar WSN cannot fulfill all the application requirements such as insect and plant disease detections. Thanks to the advanced of low cost CCD camera, a scalar WSN node may be equipped with a camera to implement low cost

WMSN node. Due to the richness of the data generated by images and the advance image processing techniques, insect and plant disease detections may be achieved. Nowadays different academic and commercial WMSN nodes are available: MeshEye, WiCa, MicrelEye, Cyclops, CITRIC, Stargate, CMUcam3, IMote2, eCAM, FireFly Mosaic. These WMSN nodes can be classified into two types: Low performance WMSN node and Medium performance WMSN node.

3.1.2.3. Low performance WMSN node

Low performance WMSN nodes, such as MeshEye, WiCa, MicrelEye, Cyclops, CMUcam3, eCAM and FireFly Mosaic, are all based on low performance microprocessor (CPU clock frequency < 100 MHz), low bandwidth wireless access medium and simple operating system. Table 3-2 provides key features of all the low performance WMSN nodes mentioned before.

Table 3-2 Key Features of Low performance WMSN nodes

Platform	Processor	RAM	Flash	Radio
Cyclops	8-bit ATmega128L MCU + CPLD	64 KB	512 KB	IEEE 802.15.4
FireFly Mosaic	60MHz 32-bit LPC2106ARM7TDMI MCU	64 KB	128 KB	IEEE 802.15.4
eCam	OV 528 serial-bridge controller JPEG compression only	4 KB (Eco)	-	RF 2.4 GHz 1Mbps
MeshEye	55 MHz 32-bit ARM7TDMI based on ATMEL AT91SAM7S	64 KB	256 KB	IEEE 802.15.4
WiCa	84 MHz Xetal SIMD Processor + 8051 ATMEL MCU	1.79 MB +128KB DPRAM	64 KB	IEEE 802.15.4
MicrelEye	8-bit ATMEL FPSLIC (includes 40k Gate FPGA)	36 KB + 1 MB external SRAM	-	Bluetooth
CMUcam3	60 MHz 32-bit ARM7TDMI based on NXP LPC2106	64 KB	128 KB	-

From Table 3-2 we can find that most of these low performance WMSN nodes are also based on one single microcontroller. Even WiCa has two cores, it still lack the mutual real time checking and validation between cores.

3.1.2.4. Medium performance WMSN node

Medium performance WMSN nodes, such as CITRIC, Stargate and IMote2, have more powerful microprocessor. Their CPU clock frequency can be higher than 400 MHz. They have enough memory resource to run an embedded Linux operating system. Table 3-3 provides key features of medium performance WMSN nodes.

Table 3-3 Key Features of Medium performance WMSN nodes

Platform	Processor	RAM	Flash	Radio
Imote2	416 MHz 32-bit PXA271 XScale processor	256 KB SRAM + 32MB SDRAM	32 MB	IEEE 802.15.4
Stargate	400 MHz 32-bit PXA255 XScale CPU	64 MB	32 MB	IEEE 802.11 and IEEE 802.15.4
CITRIC	624 MHz 32-bit Intel XScale PXA270 CPU	64 MB	16 MB	IEEE 802.15.4

Note that most of the current WMSN is based on low bandwidth wireless access medium (IEEE802.15.4), except the MEMSIC Stargate system may be equipped with multiple wireless communication transceivers. The MEMSIC Stargate boards can have an operational IEEE802.11 card along with an interfaced MICAz mote that follows the IEEE802.15.4 standard.

The number of channels, power restrictions, and channel structure are different in IEEE802.11 and IEEE802.15.4. User must choose which of the several available transceiver designs and communication protocol standards may be used to optimize the energy saving and the quality of the resulting communication.

From Table 3-3 we can find that all the media performance WMSN nodes are still based on one microcontroller. No other core in those nodes can help to make mutual real time checking and validation between cores.

3.1.3. WSN Applications

WSN is an emergent and multidisciplinary science, which is very active and competitive research field. WSNs have unlimited potential applications (air, underground and underwater): environmental data collection, smart home, smart care etc. WSN is considered as a key technology of the 21st century and as the foundation of Pervasive computing, Mobile computing, Wearable computing (Body Area Network ‘BAN’ etc.) and Internet of Things ‘IoT’. In fact, in spite of its short research history, WSN will change the service modes in the fields of remote surveillance, control and assistance, and thus bring huge impacts on the economic and social benefits. Here, we discuss some several particular kinds of applications.

3.1.3.1. Precision Agriculture

As projected in a report by United Nations, the population of the world will increase to above 9 billion in the middle of the century, and will instead keep growing and may hit 10.1 billion by the year 2100 (United Nations, 2013). Due to the increased demand of food, people are trying to put extra efforts and special techniques to increase the food production by preserving environment. Precision agriculture, which is a farming management concept based on observing and responding to intra-field variations, is one of such efforts.

Chapter 3. Dependability of Wireless Sensor Networks

Precision agriculture is about whole farm management with the goal of optimizing returns on inputs while preserving resources. Precision agriculture aims to optimize field-level management with regard to:

- Crop science: by matching farming practices more closely to crop needs (e.g. fertilizer inputs);
- Environmental protection: by reducing environmental risks and footprint of farming (e.g. limiting leaching of nitrogen);
- Economics: by boosting competitiveness through more efficient practices (e.g. improved management of fertilizer usage and other inputs).

Precision agriculture also provides farmers with a wealth of information to:

- build up a record of their farm;
- improve decision-making;
- foster greater traceability
- enhance marketing of farm products
- improve lease arrangements and relationship with landlords
- enhance the inherent quality of farm products (e.g. protein level in bread-flour wheat)

WSN nodes are used for collecting information about physical and environmental attributes whereas actuators are employed to react on the feedback to have control over the situations. Agriculture domain poses several requirements that are following:

- Collection of weather, crop and soil information
- Monitoring of distributed land
- Multiple crops on single piece of land
- Different fertilizer and water requirement to different pieces of uneven land
- Diverse requirements of crops for different weather and soil conditions
- Proactive solutions rather than reactive solutions.

Above requirements entail parallel and distributed application and processing. In addition, wireless sensors and actuators are required to collect the requisite information and to react on different situations. Decision support imposes the requirement to have processed information rather than raw sensor data.

To cope-up with such requirements, wireless sensors, actuators and their networks present themselves as a strong candidate for development of system for context acquisition, presenting acquired data to remote decision support systems and thus providing a controlled environment based on decision (Baggio, 2005; Kaemarungsi, 2012; Keshtgari & Deljoo, 2012; N. Medrano & S. Celma, 2006; Sutar, Jayesh, & Priyanka, 2012).

3.1.3.2. Smart Parking

Parking is a universal problem in most metropolitan areas that already suffers from heavy traffic congestion and air quality degradation. Limited parking space and the lack of information on parking availability make the parking search time unreasonably long. This undesirable parking search traffic leads to additional congestion, air pollution and driver frustration. Increasing parking space is discouraged by the limited land space and its high cost in urban area. Therefore, Parking Guidance and Information System (PGIS) is introduced to minimize the parking search traffic (Teodorović & Lučić, 2006).

In the PGIS, low-cost WSN nodes can be deployed into each parking slot to detect the state of the parking slot. Beyond the free parking state, the WSN node can also collect other information such as air pollution, environmental noise, etc. All the data will send to center server through Edge Router. The real-time free parking slot maps can ease the citizens parking their cars. The environmental information can great help to build a smart parking place with better air condition. Furthermore, all the data can be stored for further study.

3.1.3.3. Smart Irrigation

The key resources for plant growing are water, soil, air, sunlight and temperature. In many planting scenarios, water is indispensably controllable resource and it has a very important impact on eco-environment. A suitable irrigation schedule improves plant growing and minimizes resource consumption, while an over-irrigation induces the over-fertilizer and over-pesticide that result in polluting groundwater. However, an existing problem for many farmers (especially for those in third world) is a lack of correct knowledge and tools to practice the suitable irrigation schedule.

Thus, a new irrigation technology needs to be developed, and it needs to be reliable, adaptable, low-price and easy-to-used. Moreover, water is an increasingly scarce resource because of climatic, polluted and politicized reasons. To have a better irrigation technology that maximizes watering efficiency will be increasingly important for many countries to achieve both environmental and economic sustainability.

3.1.3.4. Smart Care

The medical device in smart care can be divided into two types: wearable and implanted. Wearable devices are used on the body surface of a human or just at close proximity of the user. The implantable medical devices are those that are inserted inside human body. There are many applications for different type of smart care, e.g. body position measurement and location of the person, overall monitoring of elderly people and ill patients in hospitals and at homes.

The wireless medical devices can provide real-time, long-term, remote monitoring for elderly people and ill patients. Due to the small smart and wearable device, they can provide similar safeguard as existing medical practices and technology with minimum distribution.

Therefore, WSN architecture for smart care can greatly help to improve the everyday life quality of elderly people and ill patients.

3.1.3.5. Industrial Control

Traditionally, applications in industrial environments are based on wired communication solutions. However, recently, the industry has shown interest in moving part of the communication infrastructure from a wired to a wireless environment, in order to reduce costs related with installation, maintenance and scalability of the applications. In this context, WSN actually represent the best candidate to be adopted as the communication solution for the last mile connection in process monitoring and control applications in industrial environments. Among many advantages, the absence of a wired infrastructure enables WSN to extract information in a simpler way than traditional monitoring and instrumentation techniques (Desai, Jain, & Merchant, 2010; Peng, Huijin, Lei, Zhi, & Anke, 2006).

3.1.3.6. Internet of Things and Web of Things

Thanks to 6LoWPAN (Y. Chen et al., 2011; Montenegro, Kushalnagar, Hui, & Culler, September 2007), RPL (IETF, 2012) and HTTP, the interoperability of WSN nodes over internet is solved. The 6LoWPAN/IPv6 allows native connectivity between WSN and Internet, enabling smart objects to participate to the Internet of Things (IoT). The evolution of IoT - the next huge opportunity - which will both attempt to connect these existing systems and then augment that by connecting more things, thanks to wireless sensor networks (WSN) and other technologies.

The Web of Things (WoT) is a vision inspired from the Internet of Things where everyday devices and objects, i.e. objects that contain an embedded device or computer, are connected by fully integrating them to the Web. Unlike in the many systems that exist for the Internet of Things, the Web of Things is about re-using the Web standards to connect the quickly expanding eco-system of embedded devices built into everyday smart objects. Well-accepted and understood standards and blueprints (such as URI, HTTP, REST, Atom, etc.) are used to access the functionality of the smart objects. These ensure the loose-coupling of services provided by the smart objects, furthermore they offer a uniform interface to access and build on the functionality of smart objects.

3.1.3.7. Summary

In this section, we discussed several kinds of WSN applications in different field. WSNs have unlimited potential (huge applications: air, underground and underwater). WSNs will be the next IT revolution. However, one of the main obstacles on the way of WSN spreading is dependable. The next part will discuss some dependable challenges in designing of WSN.

3.2. Major Dependable Challenges

Here we discuss several challenges we will meet in the process of designing a robust WSN.

3.2.1. Application Requirement

Many WSN applications, such as smart care, industrial control, smart irrigation etc., have stringent dependability (reliability and availability) requirements, as a system failure may result in economic losses, put people in danger or lead to environmental damages. Moreover, WSN nodes need to work in harsh environments twenty-four hours per day, seven days per week. Therefore, these real world requirements demand a great deal of requirement on dependability.

3.2.2. Dependability Threats

These are many threats can affect the dependability of real world WSN application. These threats can be divided into two main classes:

- transient faults implicate that the sensor recovers its normal behavior when e.g., the system is reset or the fault stimulus ceases,
- permanent faults inflect defects that have a permanently effect.

Here we introduce several threats to the dependability of the overall real world system.

- Degradation of the battery
- Different temperature responses in the processor and radio oscillator, which causes numerous network failures
- Water infiltrations, which introduce degradations within the hardware
- Physical damage
- Communication faults (e.g., interference, multi-path fading, noises)
- Direct sunlight that swamps the sensor infrared signal
- High-energy particle that corrupts the memory (SRAM) of WSN node
- Software bugs, memory leaks, memory corruptions and pointer-initiated memory violation

3.2.3. Resource Constraint

The WSN nodes forming a network suffer from the limitations of several resources, such as storage, CPU, bandwidth, communication, sensing, and battery power (or energy). In particular, energy is the most crucial resource as it determines the lifetime of the sensors and hence the lifetime of the entire network. Energy poses a serious problem for designers, because in the real world deployment, it is very difficult some application may impossible to access the sensors and recharge or renew their batteries. Furthermore, when the energy of the sensors decreases to a certain threshold, they become unreliable (or faulty). They may not be able to function properly. Consequently, the behavior of those faulty sensors will have a major impact on the network performance. Thus, network protocols and algorithms designed to be run by the sensors should be as energy efficient as possible to extend their lifetime and hence prolong the network lifetime while guaranteeing good performance overall.

3.3. Current Dependable Approaches

3.3.1. Current Approaches

Current dependable approaches for WSN are based on faulty sensor nodes. Due to the resource constraint, traditional dependable approaches such as processor instruction error detection (Lipetz & Schwarz, 2011), processor instruction retry (Spainhower & Gregg, 1999; Steve Bostian, 2012), ECC protection memory (Dell, 1997), memory sparing (Hewlett-Packard Development Company, 2010), redundant I/O (Intel Corporation, 2013b; Oracle, 2010), I/O partitions (IBM, 2011) and RAID disk storage (P. M. Chen, Lee, Gibson, Katz, & Patterson, 1994) cannot be directly applied in WSN field.

Therefore, currently dependable approaches mainly focus on improving the reliability of the whole network. Their goals are trying to carry on the overall task of the network even some WSN nodes are in fault status. These fault tolerant techniques are based on the spatial redundancy (Gao et al., 2007; Hsieh et al., 2010) or spatial and time redundancy (Khan et al., 2012; M.-H. Lee & Choi, 2008) of WSN network. They are implemented on MAC Layer (W. L. Lee, Datta, & Cardell-Oliver, 2006), transport layer (Jones & Atiquzzaman, 2007; Sankarasubramaniam, Akan, & Akyildiz, 2003), routing protocol (Akkaya & Younis, 2005; Al-Karaki & Kamal, 2004) and middleware (Yan, Chang, Qin, Li, & Liu, 2013).

In all those approaches, the WSN nodes are still based on only one core, and they are not reliable. This dissertation focuses on developing a more reliable WSN node by introducing multicore architecture to improve the reliability of every single node. Through this mechanical, the reliability of whole network is also involuntary improved.

3.3.2. TMS570 Safety MCU

The TMS570 devices are the industry's first Cortex™ ARM® R4 and Cortex™ ARM M3 based MCUs, targeting safety critical and driver assistance automotive applications. TI offers TMS570 with a patent pending implementation of the lock-step Cortex ARM R4 cores on a single device as well as dual core offerings of Cortex ARM R4 plus Cortex ARM M3 on a single device. The TMS570 multi-core devices offer performance, safety and rich peripheral MCU integration such as timers, ADC, CAN, and FlexRay™ (Texas Instruments Incorporated., 2013).

The Hercules™ TMS570 Safety MCU family enables customers to easily develop safety-critical products for transportation applications.

Developed to meet the requirements of ISO 26262 ASIL D and IEC 61508 SIL 3 safety standards and qualified to the AEC-Q100 automotive specification this ARM® Cortex™-R4 based family offers several options of performance, memory and connectivity. Dual core lockstep CPU architecture, hardware BIST, MPU, ECC and on-chip clock and voltage monitoring are some of the key functional safety features available to meet the needs of automotive, railway and aerospace applications.

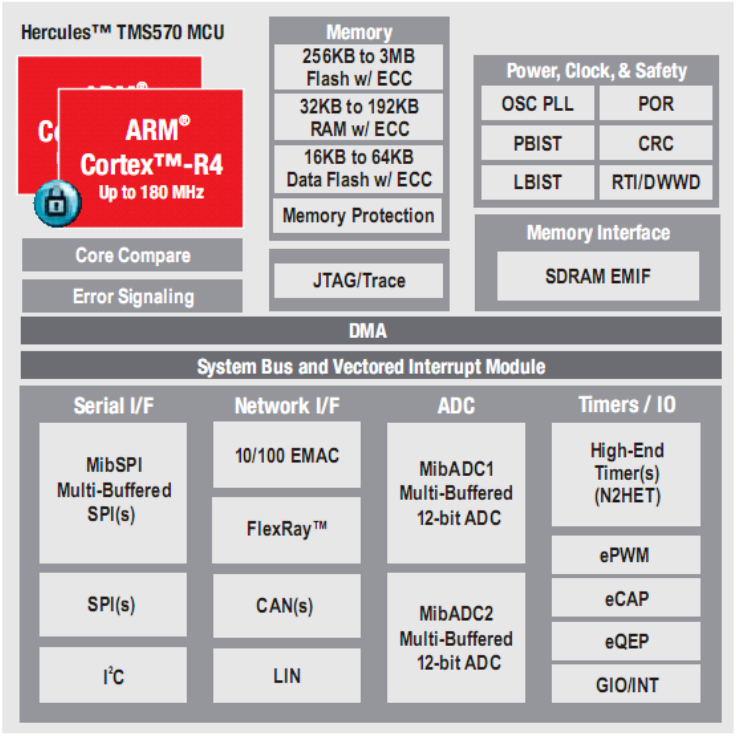

Figure 3-6 Block diagram of TI TMS570 microcontroller (Texas Instruments Incorporated., 2013)

However, the TMS570 is design for safety critical application, but the power consumption is not optimal enough. Therefore, the TMS570 is not the best microcontroller for WSN node.

3.4. Observations of Real World WSN Deployments

As far as we know, the single core WSN node in real world deployments is not reliable. About 10% to 20% of nodes fail to join the network in first week. We met this type of problem in our Hydrasol project and Single Core Module (SCM) deployment. Our collaboration partner in Irstea also informed us similar result on the deployment of Libellium WSN node. The WSN deployment on Great Duck Island by UC Berkeley also suffered about 50% node failure within 4 days (Joseph Polastre, Szewczyk, Mainwaring, Culler, & Anderson, 2004). The SensLAB project (SensLAB team, 2013) also suffered this type of fault in SensLAB testbed. Kaemarungsi (2012) mention the same problem when they deploy WSN node in sugarcane field in Thailand.

The failure of WSN real world deployment may due to many reasons. Somehow, the high-speed ultra-low power CMOS technology adapted in WSN nodes also increased the failure rate. When WSN implemented by lower power supply voltage, the power consumption can be lower, but meanwhile, the MTTF of chip also decease (Maheshwari, Burleson, & Tessier, 2004). The lifetime of chip decreases by a factor of 2.2 for every 10℃ increase in operating temperature (Zhang & Orshansky, 2008). The failure rate of chip significantly increases when the technology node size decreased (Borkar, 2005). Nightingale, Douceur, and Orgovan (2011) shows the crash probability will increase by a factor of 100 after a machine has crashed once. In addition, the probability continues to increase with subsequent crashes.

Besides, at least in our Hydrasol project and SCM deployment, the watchdog is already active. However, the observed results show that the watchdog did not make those lost nodes rejoins the network. Therefore, we concluded that the single core node is not reliable and the watchdog is not efficient to recovery from this type of fault. The causes of the WSN (LiveNode and SCM) soft errors are unknown. My work will focus on the development of an integrated multicore platform (WSN node, Hardware support, fault injection testbed) which enables to implement energy efficient and robust multicore modular WSN node and to ease the debug, test and validation. Moreover we hope that this integrated platform will enable to understand precisely and accurately the reason of the soft errors and to recover from failure.

3.5. Summary

In this chapter, we discussed the different WSN nodes, WSN application, dependability threats and current approaches. Current approaches adopt symmetric space and time redundancies, which are not appropriate for high-energy constraint and resource context aware concept. In this dissertation, we will investigate dissymmetric multicore WSN node architecture, which will meet both energy consumption constraint and resource context-aware concept to improve the robustness and the lifetime of WSN node.

Chapter 4. Multicore WSN Node Architecture

This chapter gives an overview of multicore WSN architecture and specifies some interesting technical details.

To implement a long lifetime WSN node powered by standard battery, currently an ultra-low power single core (8, 16 or 32-bit) is used. However this implementation solved partially energy consumption problem but it still not meet WSN node robustness requirement. My work focused on the research and development of a new WSN node architecture aiming to increase at the same time the WSN node lifetime, modularity and robustness. If we can achieve these objectives the new WSN node will meet the requirements of high constraint indoor (smart care e.g.) and outdoor (precision agriculture e.g.) applications.

Therefore, in order to fulfill the requirements, we will present a new energy efficient multicore WSN architecture, which can highly improve the reliability and safety without sacrificing simplicity. The rest part of this chapter will provide more detail on this new architecture.

4.1. Introduction

As we mentioned in Table 3-1, Table 3-2 and Table 3-3 before, there are many WSN/WMSN nodes, such as MICAz, MICA2, Imote2, TelosB, IRIS and Cricket are available in the shelf. These WSN nodes are quite similar in term of functionality. They are based on one microcontroller equipped with a unique wireless access medium having 200m LOS range. Among these platforms, the most common WSN research software and hardware platform are TinyOS (Berkeley, 2013; Levis, 2006) and Tmote Sky or TelosB (J. Polastre, R. Szewcyzk, C. Sharp, & D.Culler, 2004; J. Polastre, Szewczyk, & Culler, 2005), developed by UC Berkeley's teams. Figure 4-1 shows the diagram of TelosB. TelosB has a Texas Instruments MSP430 microcontroller and a Chipcon AS (acquired by TI) IEEE 802.15.4-compliant radio. The power consumption of TelosB is almost one-tenth of previous mote platforms while providing greater performance and throughput. It eliminates programming and support boards, while enabling experimentation with WSNs in lab, testbed, and deployment settings.

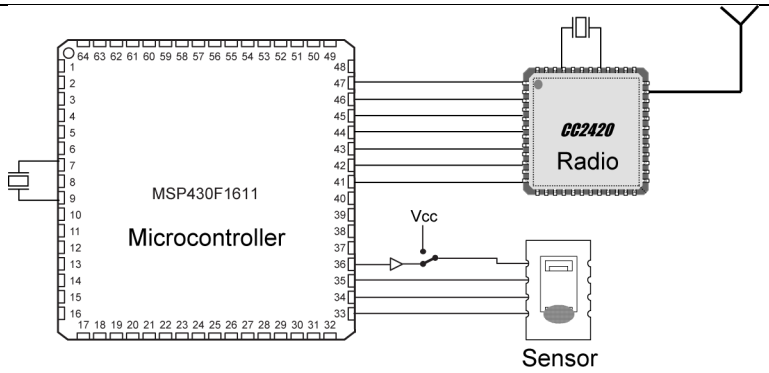

Figure 4-1 Block diagram of TelosB

From Figure 4-1 we can find easily that if the microcontroller or radio transceiver suffers some faults, nothing else can help to recover. Moreover, the outdoor environment is very complex and some sensors need constant voltage power supply.

These existing WSN nodes are not designed to fit the requirements of outdoor applications. They are not robust, configurable and flexible to meet the requirements of high reliability. Therefore, we present a new first fault tolerant and configurable WSN node architecture based on multicore with very low energy consumption. The new multicore architecture provides high performance, more flexibility, while maintaining a small form factor. It allows user to develop software to utilize the features of the multicore architecture to improve the reliability of users' application.

4.2. Multicore WSN Node Architecture

4.2.1. Generalized Multicore Architecture

Figure 4-2 presents the block diagram of multicore architecture. There are three types of cores in the node.

- The Main App Core is a normal application core as same as in single core WSN node.
- The Auxiliary Core is optional core; the function of this core is depended on specific application.
- The FD & FR Core is the key component in the multicore architecture. It coordinates all components in the nodes, runs as a monitor of Main App Core, detects faults in the Main App Core. It will isolate the faulty Main App Core and active the Auxiliary Core to substitute the Main App Core if necessary. Through the switching of core, multicore WSN node can provide seamless services even in the presence of faults.

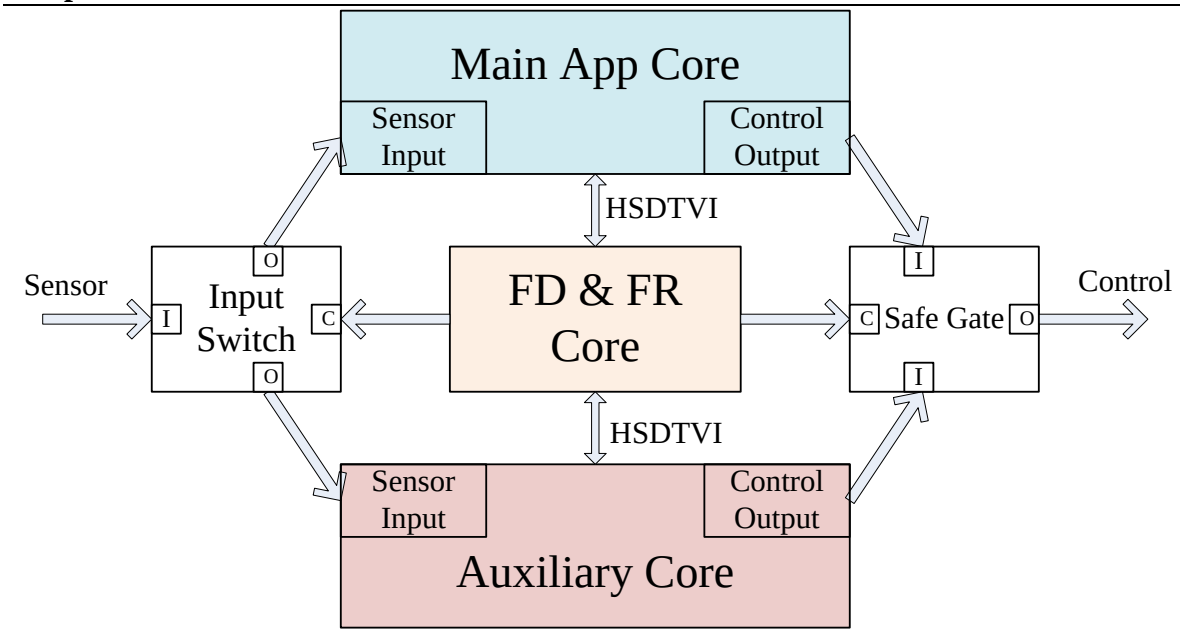

Figure 4-2 Block diagram of Multicore Architecture

The Input Switch and Safe Gate are controlled by the FD & FR Core. So FD & FR Core can isolate the fault Core from Sensor Input and Control Output. This can greatly help to achieve a functional safety system.

4.2.2. Functional Safety Mechanism

Functional Safety is the part of the overall safety of a system or piece of equipment that depends on the system or equipment operating correctly in response to its inputs, including the safe management of likely operator errors, hardware failures and environmental changes.

In IEC 61508, Functional Safety's definition is: Safety is the freedom from unacceptable risk of physical injury or of damage to the health of people, either directly or indirectly as a result of damage to property or to the environment. Functional Safety is part of the overall safety that depends on a system or equipment operating correctly in response to its inputs.

In ISO 26262, Functional Safety's definition is: Absence of unacceptable risk due to hazards caused by mal-functional behavior of electrical and/or electronic systems

Multicore Architecture can greatly help to achieve functional safety through the active FD & FR Core. When the FD & FR Core detects fault, it can control the Safe Gate to ensure the safety of whole system.

The FD & FR Core is independently running aside APP Core, so the detection and recovery or isolate process will never be interfered by the application. The separation can also increase the reliability of detection and recovery or fault part isolation process.

4.2.3. Fault-tolerant Mechanism

For outdoor and reliable applications such as environmental data collection and smart care, the robustness is a key constraint for large-scale WSN deployment. In multicore WSN node, it is possible to implement *Standby sparing* (space redundancy) for fault tolerant approaches.

Standby sparing is a scheme for active hardware redundancy as shown in Figure 4-3. Only one of *n* modules is operational and provides the system's output. The remaining *n*-1 modules serve as spares.

A *spare* is a redundant component, which is not needed for the normal system operation. A switch is a device that monitors the active module and switches operation to a spare if an error is reported by fault-detection unit FD.

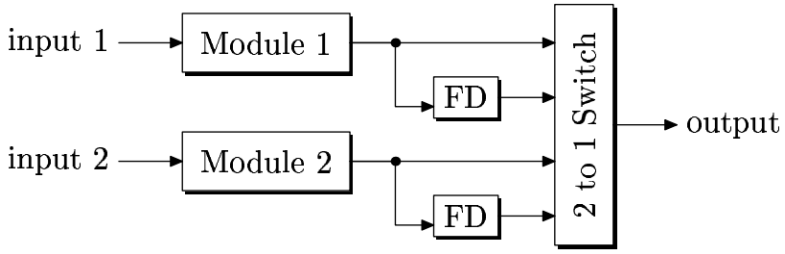

Figure 4-3 Standby sparing system (Dubrova, 2013)

It is very easy to find that multicore architecture is a two modules standby sparing system. Therefore, the multicore WSN node can continually provide service even one modules is in the present of faults. Only until both modules meet fault, the system will stop operation. This enables to implement robust WSN for critical applications.

4.2.4. Resource-aware Mechanism

In this section, we will show that multicore architecture is energy efficient. In general, a WSN node will have the following components or layers:

- Application software
- Middleware
- Communication and administration protocols
- Real-time operating system
- Hardware

Since the boundary between middleware and communication & administration protocols is not clearly defined, we may thus consider that a wireless sensor has only four main

components: application software, communication protocol, real-time operating system and hardware.

The energy consumption (lifetime) is the key constraint of WSN. Thus to minimize energy consumption, it has obviously to optimize the resource consuming of each component of a WSN node, but this approach is still not efficient enough to meet the requirement of WSN application lifetime. Consequently, the cross layering approach is generally adopted. To have a one-year lifetime, a WSN node equipped with a cell battery must consume less than $100\,\mu W$. In fact, the wireless communication is the energy consuming behavior, in some case, which may consume 75% of total energy of an application. The wireless communication energy consuming may be estimated approximately by (4.1):

$$\xi(e) \approx m_s * n_h * \xi_b + \xi(t_r) \tag{4.1}$$

where m_s: message size; n_l: hop number; ξ_b: energy for sending 1 bit and $\xi(t_r)$: energy for listening or receiving message (duration t_r).

For example, from (4.1) different approaches may be applied to minimize energy consuming in different layers for the single core wireless node:

- Routing protocol:
  - shortest (optimal) path to minimize the hop number,
  - data fusion or data aggregation to minimize the message size,
- Operation mode: entering sleep &wakeup state to minimize the listen time. Notice that with the current wireless access medium (e.g. IEEE802.15.4) the listening or receiving message consumes more energy than sending message.

All these previous approaches are necessary to increase WSN lifetime. However, from our point of view, it is essential to investigate context-aware particularly resource-aware issue to minimize energy consumption. Thus it seems important to implement a multicore WSN node, which having different computation capacities to be able to fully explore the resource-aware approach. With a single core WSN node, it is not energy efficient because the node system will run with the same frequency for any kind of tasks (time or not time constraint). Therefore, as the unicore WSN node system, it is too powerful for a simple task application, but not powerful enough for the complex one. Comparing with multicore system, the unicore WSN node system has further execution time and higher energy consumption. Moreover, on one hand, with more powerful CPU the message may be compressed to minimize its size. On the other hand, more powerful computation resource (CPU and memory) enable to implement environment estimator to decrease the sample frequency (minimize communication traffics).

The task computation energy may be quantified by (4.2):

$$\xi_i \approx a_i * \rho_i * t \tag{4.2}$$

where t: execution duration, a_i: is a set of instructions, i task constant which depends on the size and the complexity of task and ρ_i is the necessary power to execute one instruction i.

For a unicore node the energy consumption of an application is:

$$\xi(A) = \sum_{i=1}^{N} \xi_i \tag{4.3}$$

where A is an application having N tasks.

In case of multicore node the energy consumption of an application is:

$$\xi^M(A) = \sum_{i=1}^{K} \xi_i^1 + \cdots + \sum_{i=1}^{L} \xi_i^P \tag{4.4}$$

where $K + \cdots + L$ $(N = K + \cdots + L)$ is the task number of the application, P is the core number and $\sum_{i=1}^{K} \xi_i^1$ is the energy consumption of core 1 having K tasks.

Thus in case of multicore sensor node, a task may be allocated to a core by taking into account its energy consumption (allocation with energy efficient as objective function). If one of the single core wireless task may be executed by another core consuming less energy than the single core one, thus:

$$\xi(A) > \xi^M(A) \tag{4.5}$$

For the multicore node, it needs to implement an efficient power management mechanism, which enables to switch off the unused cores. In the following chapter, the detail of the implementation will be presented.

4.2.5. Dissymmetrical Multicore Structure

In fact general-purpose fault tolerant system, such as high performance computer or critical control system like fly-by-wire systems in aircraft, space and time redundancy are based on symmetric cores because these systems do not have high resource constraint. However, the space and time symmetric fault tolerant system concept is not appropriate for implementing WSN node where energy consumption is one of the most important features. Therefore, reducing power consumption and cost are increasingly across all segments of product. Users want improved robustness, battery life, size, and cost for WSN nodes.

The robustness requires the fault detection, test and validation based on multicore. The traditional symmetrical multicore structure may improve the robustness, but the total cost and power consumption will significantly increase and beyond the acceptance range.

To meet these requirements, dissymmetrical multicore structure will be an essential element that must to be adopted. In dissymmetrical multicore structure, comparing with Main

App Core, the FD & FR Core will be a smaller, lower cost, lower performance core that consume much less power. Though dissymmetrical multicore structure will bring a little bit software design complexity, it can help to improve the multicore architecture in all four of these vectors: robustness, power, cost and size.

In this dissertation, we will evaluate different type of cores and build dissymmetrical multicore structure based on these cores.

4.3. Different Type of Cores

There many technical decisions need to consider when we implement multicore architecture. One of the main tasks is the choices of different cores. Here we briefly introduce some microcontrollers used in our design.

4.3.1. IGLOO nano FPGAs

IGLOO® nano FPGAs is a low-power FPGA from Actel (acquired by Microsemi). IGLOO® nano low-power FPGAs offer groundbreaking possibilities in power, size, lead-times, operating temperature, and cost. Available in logic densities from 10,000 to 250,000 gates, the 1.2 V to 1.5 V IGLOO nano devices have been designed for high-volume applications where power and size are key decision criteria. Priced competitively in the market, IGLOO nano devices are perfect ASIC or ASSP replacements, yet retain the historical FPGA advantages of flexibility and quick time-to-market in low-power and small footprint profiles (Microsemi, 2013) (Actel-Corporation, 2009).

They Features of IGLOO® nano FPGAs are:

- Ultra-low power in Flash*Freeze mode, as low as 2 µW
- Variety of small footprint packages as small as 3x3 mm
- Zero lead time on selected devices
- Known good die supported
- Enhanced commercial temperature
- Reprogrammable flash technology
- 1.2 V to 1.5 V single voltage operation
- Enhanced I/O features
- Clock conditioning circuits (CCCs) and PLLs
- Embedded SRAM and nonvolatile memory (NVM)
- In-system programming (ISP) and security.

We mainly use the IGLOO nano FPGAs as the configurable network connector for the devices on board. With the configurable IGLOO, the circuit can change the connections between cores; adjust work states of all cores without making any wired change. The IGLOO family of flash FPGAs, based on a 130-nm flash process, offers the lowest power FPGA, a single-chip solution, small footprint packages, reprogram ability, and an abundance of advanced features. The Flash*Freeze technology used in IGLOO devices enables entering and exiting an ultra-low-power mode that consumes nano power while retaining SRAM and register data. Flash*Freeze technology simplifies power management through I/O and clock management with rapid recovery to operation mode. The Low Power Active capability (static idle) allows for ultra-low-power consumption while the IGLOO device is completely functional in the system. This allows the IGLOO device to control system power management based on external inputs (e.g., scanning for Passive Infrared Motion Detector output stimulus) while consuming minimal power.

4.3.2. 4-bit NanoRisc

The NanoRisc is an ultra-low power 4-bit microcontroller coming in a small 8-pin SO package and working up to 0.4 Million Instructions Per Second (MIPS). It consumes only 5.8 µA in active mode and 3.3 µA in standby mode. On the contrary, ATMEGA1281 needs 500 µA in active mode and 130 µA in standby mode (Atmel-Corporation, 2012b). Base on the ultra-low power feature of NanoRisc, it can greatly help to improve the lifetime of WSN node when node works in Sleep &Wakeup mode. Moreover, it requires no external component, so it is very easy to integrate to a multicore WSN node design. The NanoRisc contains the equivalent of 8 kB of Flash memory and a RC oscillator with configurable running frequency from 32 to 800 kHz. It also has an integrated 4-bit ADC, a power-on reset, watchdog timer, 10-bit up/down counter, PWM and several clock functions. It has a sleep counter reset allowing automatic wake-up from sleep mode. It is designed for use in battery-operated and field-powered applications requiring an extended lifetime. A high integration level makes it an ideal choice for cost sensitive applications.

4.3.3. 8-bit ATMEGA1281

The ATMEGA1281 is running at 8 MHz and delivering about eight Million Instructions Per Second (MIPS) (Atmel-Corporation, 2012b). This 8-bit microcontroller has 128-Kbyte flash program memory, 8-Kbyte static RAM, internal 8-channel 10-bit analog-to-digital converter, 3 hardware timers, 48 general-purpose I/O lines, 1 external Universal Asynchronous Receiver Transmitter (UART) and one SPI port.

4.3.4. 8-bit RISC core microcontroller AVRRF

The AVRRF is an IEEE 802.15.4 compliant single chip combines an industry-leading AVR microcontroller and best-in-class 2.4GHz RF transceiver (Atmel-Corporation, 2012a). It runs at 16 MHz and delivers optimal performance 16 Million Instructions Per Second (MIPS). This 8-bit microcontroller has 128-Kbyte flash program memory, 16-Kbyte static RAM. Comparing with ATMEGA1281, AVRRF is two times faster and has two times bigger SRAM. These new features enable AVRRF to build a higher performance WSN node.

4.3.5. 32-bit RISC core microcontroller AT91SAM7Sx

The AT91SAM7Sx running at 48 MHz delivers about forty-three Million Instructions Per Second (MIPS) (Atmel-Corporation, 2011). The AT91SAM7Sx 32-bit RISC microcontroller has the following on chip devices: 512-Kbyte of flash program memory, 64-Kbyte of static RAM, 8-channel 10-bit analog-to-digital converter, three hardware timers, thirty-two general-purpose I/O lines, one USB 2.0 full speed (12 Mbps) device port, two external Universal Synchronous/Asynchronous Receiver Transmitter (USART), one I²C interface and one master/slave Serial Peripheral Interface (SPI) port.

4.3.6. Raspberry Pi Board

The Raspberry Pi Board is a credit-card-sized single-board computer developed in the UK by the Raspberry Pi Foundation with the intention of promoting the teaching of basic computer science in schools (Raspberry Pi Foundation, 2013). The Raspberry Pi board has a powerful SoC integrating three cores: Low Power ARM1176JZ-F Applications Processor, Dual Core VideoCore IV® Multimedia Co-Processor Graphics Processing Unit(GPU) and Image Sensor Pipeline (ISP). The Raspberry Pi Board runs standard Linux operating system. The Raspberry Pi Board supports different types of camera, USB and Camera Serial Interface (CSI), and WiFi module.

The Figure 4-4 shows the block diagram of the Raspberry Pi Board.

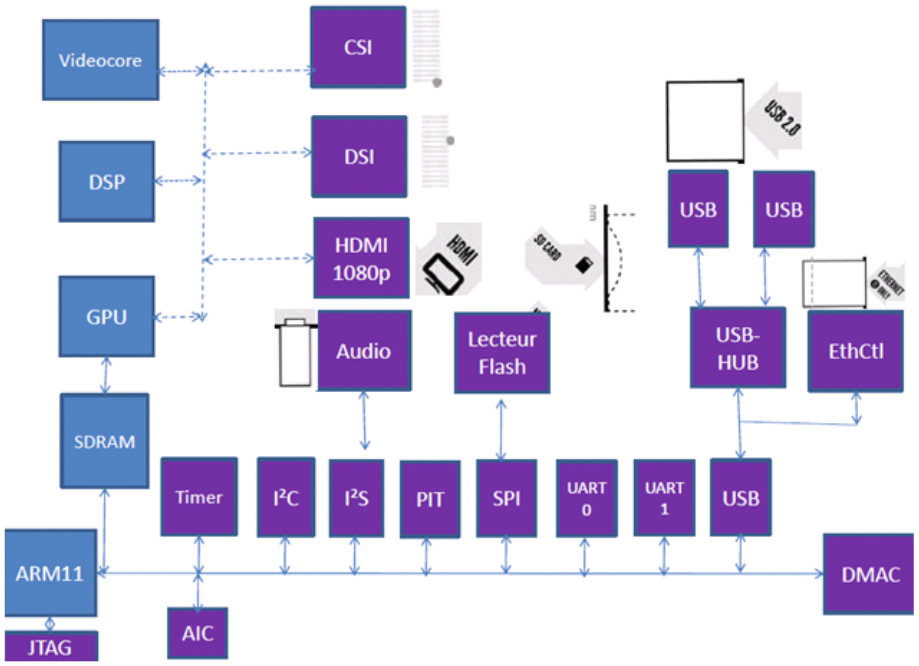

Figure 4-4 Block diagram of the Raspberry Pi Board

4.3.7. PandaBoard ES Board

The PandaBoard ES Board is a low-power, low-cost single-board computer development platform based on the Texas Instruments OMAP4460 system on a chip (SoC) (PandaBoard ES, 2013). The PandaBoard ES Board has a Dual-core 1.2 GHz ARM A9 chip with 1GB RAM. The Figure 4-5 shows the block diagram of the PandaBoard ES Board.

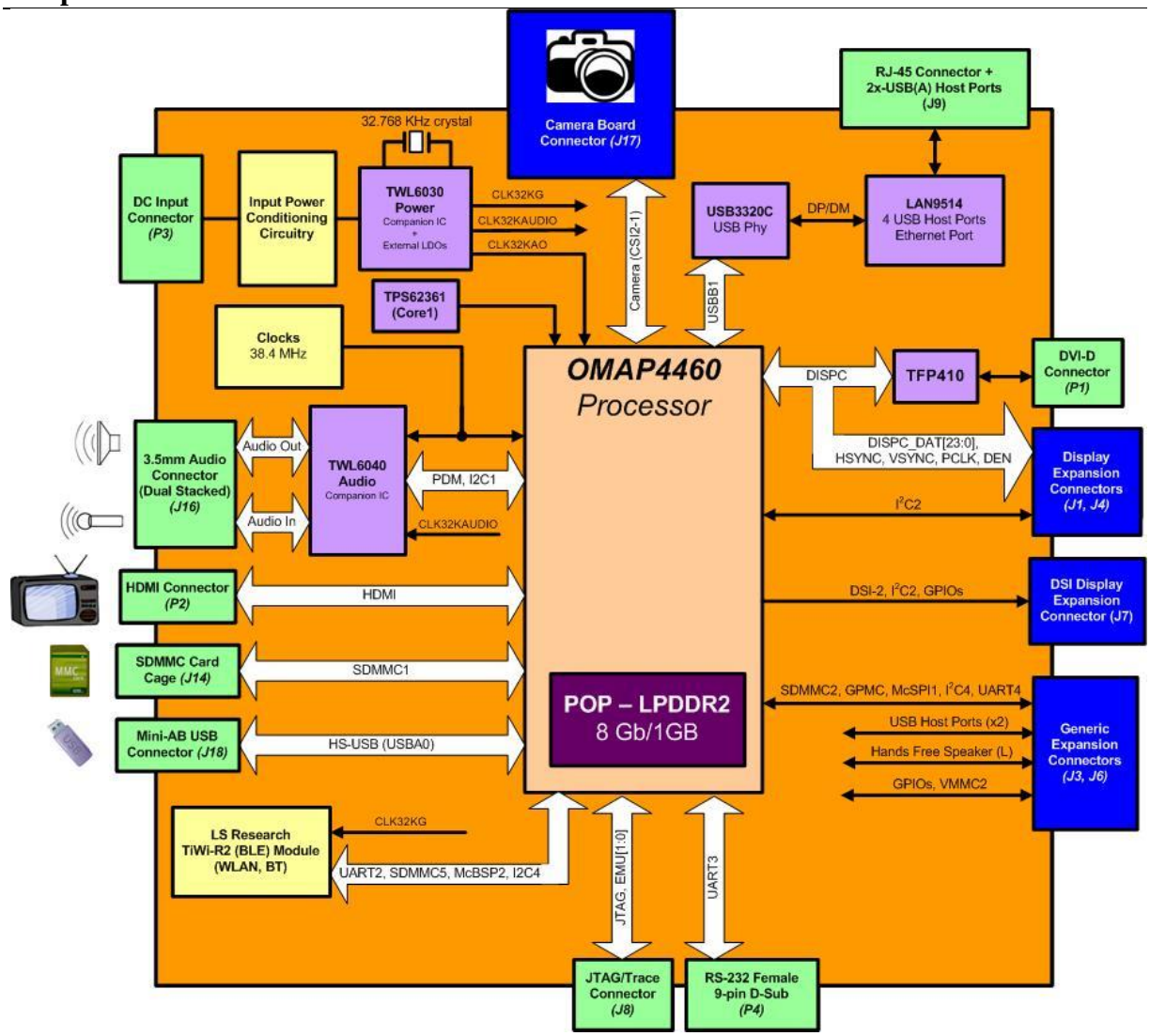

Figure 4-5 Block diagram of the PandaBoard ES Board (PandaBoard ES, 2013)

4.3.8. ARM Cortex™-M3 Based Microcontroller

The ARM Cortex™-M3 processor is the industry-leading 32-bit processor for highly deterministic real-time applications, specifically developed to enable partners to develop high-performance low-cost platforms for a broad range of devices including microcontrollers, automotive body systems, industrial control systems and wireless networking and sensors. The processor delivers outstanding computational performance and exceptional system response to events while meeting the challenges of low dynamic and static power constraints. The processor is highly configurable enabling a wide range of implementations from those requiring memory protection and powerful trace technology to cost sensitive devices requiring minimal area (ARM Ltd., 2013).

The ARM Cortex™-M3 based microcontroller becomes more and more popular in wireless networking field. Even though we did not use it in this dissertation, I think we will use it in next design.

4.3.9. Summary

The Table 4-1 provides the key features of different cores used in this dissertation.

Table 4-1 Key features of Different Core

Feature	IGLOO	NanoRisc	ATMEGA1281	AVRRF	AT91SAM7x	Raspberry Pi Board	PandaBoard ES
Bus Width		4	8	8	32	32	32
Clock Speed (MHz)	Up to 250MHz	32kHz-800kHz	8MHz	16MHz	Up to 55MHz	700MHz	1.2GHz
SRAM/SDRAM	36*1024bit	80*4bit	8 KB	16KB	64 KB	512MB	1GB
Flash/SD	1Kbit	8KB	128 KB	128KB	512 KB	Up to 32GB	Up to 32GB
VCC	1.2 V–1.5V@Core 1.2V-3.3@IO	2.3-5.5 V	1.8 V–5.5 V	1.8V-3.6 V	1.8V@Core 3.3V@IO	5V	5V
Power Consumption Active	N/A	5.8µA	3.2mA@3V 4MHz 0.5mA@2V 1MHz	2.5mA@3V 8MHz	8.4mA@3.3V 8MHz	~335mA	~450mA
Power Consumption Idle	N/A	3.3µA	0.7mA@3V 4MHz 0.14mA@2V 1MHz	0.8mA @3V 8MHz	1.06mA@3.3V 1MHz	~335mA	~450mA
Power Consumption Sleep	24µW @ Flash*Freeze	0.32µA	<5µA	1.65µA	34.3µA	N/A	N/A
FIT*	22.39	13.09	48.07	51.70	58.93	249.94	420.24

*The raw MTBF or FIT data is taken from manufacturers (Atmel-Corporation, 2012c; Kemet, 2012; Linear, 2009; Microsemi-Corporation, 2011; Onsemi, 2012).

From Table 4-1, we can find that the high perform microprocessors normally consume more energy meanwhile have higher fault rate. Therefore, we prefer to implement the FD & FR Core with lower energy consumption, simpler functionality but higher reliability microcontroller, such as NanoRisc.

4.4. HSDTVI Interface

4.4.1. Introduction

The multicore architecture is highly based on the fault detection of Main App Core and Auxiliary Core. In order to enable the efficient fault detection, we implement a specific Interface: Hardware Support Debug Test and Validation Interface (HSDTVI).

The HSDTVI Interface provides a new method to meet the basic requirements of debugging, testing and validating the hardware and software of Main App Core and Auxiliary Core. Unlike modern high performance microprocessors, which have powerful resource and debug tools; resource constrain microcontroller has limited resource; the debugging methods on microcontroller are relative simple. Historically, the following methods of debugging a microcontroller application are following:

- Printf: using a debug serial port to output string to help developer gathers the inside information. It is easy to use. However, printf through RS232 serial port is very slow, maximum speed is only 115.2kbps. The overhead, such as code, time and stack consumption of printf is heavy. Normally printf cannot use in IRQ handler. Therefore, printf is not appropriate for the real-time operating development.
- JTAG: using a JTAG emulator to examine and modify registers and memory and provide step-by-step execution. Need programmer to manually interact, very slow owing to interact. Due to the JTAG emulator, it will be very difficult to use in real world environment. Moreover, the JTAG is an efficient tool to debug sequential program but not adapt to debug concurrent program.

So current debug method needs to be improved to ease the development of Robust WSN application. New features are expected:

- Easy to use
- Can debug IRQ handler and concurrent program
- Light overhead (no side effect)
- High speed
- Can help to localize the dysfunction of an application
- Ease fail detection and recovery

Therefore, we present the HSDTVI to provide another way to debug trace and validate the microcontroller running state.

4.4.2. HSDTVI Architecture

The Figure 4-6 presents the block diagram of the HSDTVI Interface. The HSDTVI is a communication & control bus between two devices: HSDTVI Slave and HSDTVI Master.

- The HSDTVI Master receives the checkpoints from the HSDTVI Slave through the HSDTVI Interface, analyzes and monitors the state of the HSDTVI Slave. If the HSDTVI Master detects fault in the HSDTVI Slave, it can reset, reboot, or power off the HSDTVI Slave. All the receiving, checking and reacting are running in real-time.

- DataBus is a set of GPIO between HSDTVI Slave and HSDTVI Master. If the HSDTVI Slave and HSDTVI Master have enough GPIO resource, this port can use as much GPIOs as possible to get maximum debug information. If the GPIO resource is limited, this port also can decrease to only one pin.
- The Reset pin is the Reset pin of the HSDTVI Slave. If the HSDTVI Master detected fault in DUT, this pin can be used to reset the HSDTVI Slave.
- The PowerEn pin is the power control pin of the HSDTVI Slave. If the HSDTVI Master detected fault in the HSDTVI Slave, this pin can be used to power off and power on the HSDTVI Slave.
- The WR pin is used to speed up the checkpoint send speed. It is the latch clock of DataBus.
- The UART and JTAG is optional pin in the HSDTVI, reserved for the compatibility with traditional debug methods.

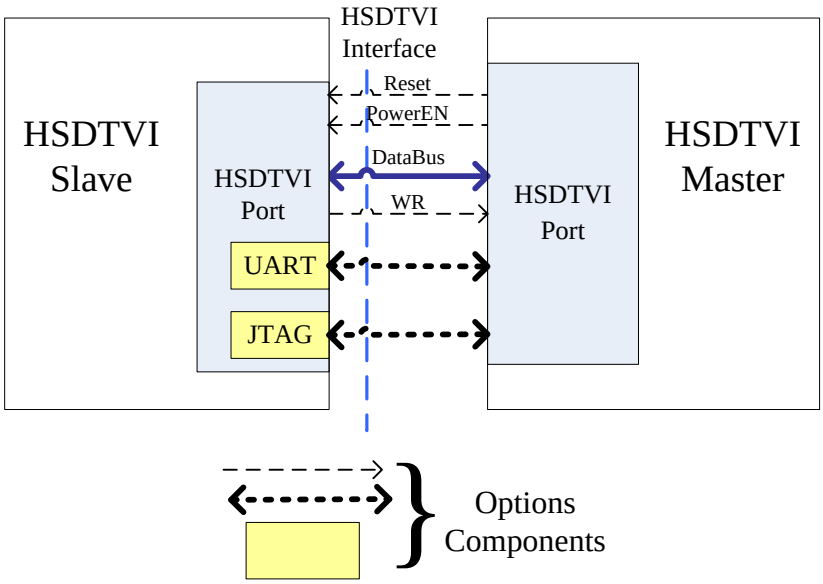

Figure 4-6 The HSDTVI Architecture

4.4.3. Different Scenario of the HSDTVI Implementation

The HSDTVI is a configurable interface. It can be mainly divided into two types of usages: Debug Mode and Real-time Fault Detect Mode (mutual debug and fault detection).

4.4.3.1. Debug Mode

The Figure 4-7 shows the HSDTVI interface used for Debug Mode Scenario. In this mode, the HSDTVI Slave is an 8-bit AVR/AVRRF microcontroller with IEEE802.15.4 wireless access media; the HSDTVI Master is a powerful microprocessor, which is much more powerful than the HSDTVI Slave. Therefore, the HSDTVI Slave can send checkpoints

frequently, and the HSDTVI Master is powerful enough to record the checkpoints and store them for further analysis.

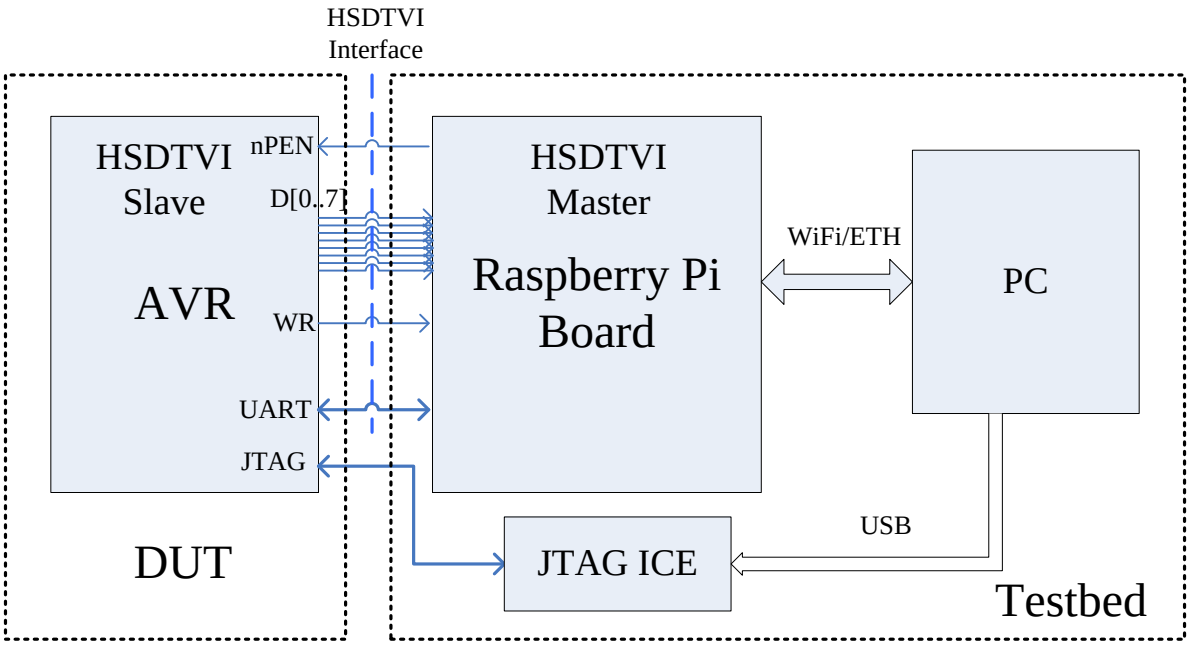

Figure 4-7 The HSDTVI Interface used for Debug Mode Scenario

Table 4-2 provides the detail pins of the HSDVTI between AVR and Raspberry Pi Board.

Table 4-2 the HSDVTI Pin connections between AVR and Raspberry Pi Board

HSDTVI Pin	**AVR/ AVRRF**	**Direction**	**RASP**	
Databus[0]	PE0	→	GEN0	GPIO17
Databus[1]	PE1	→	GEN1	GPIO18
Databus[2]	PE2	→	GEN2	GPIO27
Databus[3]	PE3	→	GEN3	GPIO22
Databus[4]	PE4	→	GEN4	GPIO23
Databus[5]	PE5	→	GEN5	GPIO24
Databus[6]	PE6	→	GEN6	GPIO25
Databus[7]	PE7	→	GCLK	GPIO4
WR	PG2	→	CE1	GPIO7
nPEN	To PSU	←	CE0	GPIO8
UART	TXD1 RXD1	↔	RXD0 TXD0	UART

Figure 4-8 shows the Circuit Board of the HSDTVI used for Debug Mode Scenario. This HSDTVI are connected between 8-bit AVR RISC in iLive[Page 86] and 32-bit ARM11 SoC in Raspberry Pi.

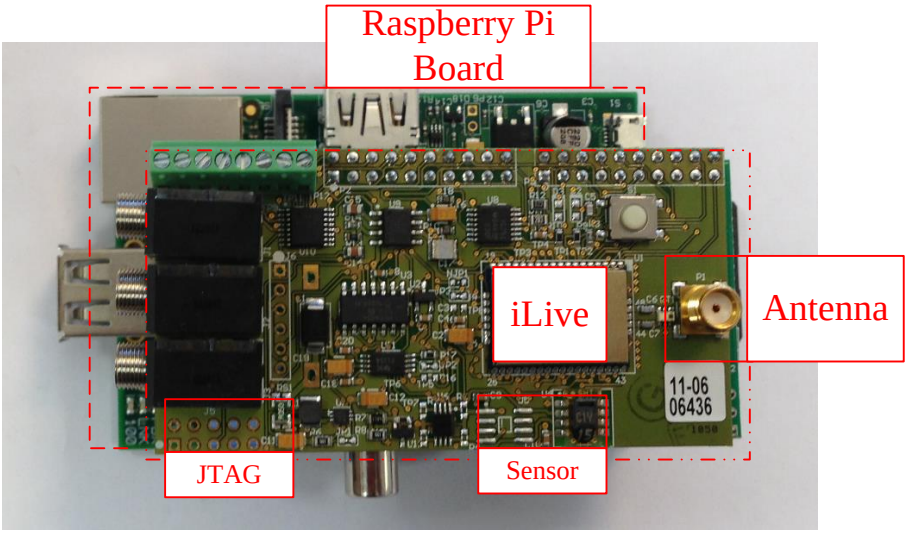

Figure 4-8 Circuit Board of HSDTVI used for Debug Mode Scenario

Figure 4-9 shows the HSDTVI Debug Trace and Validate Process.

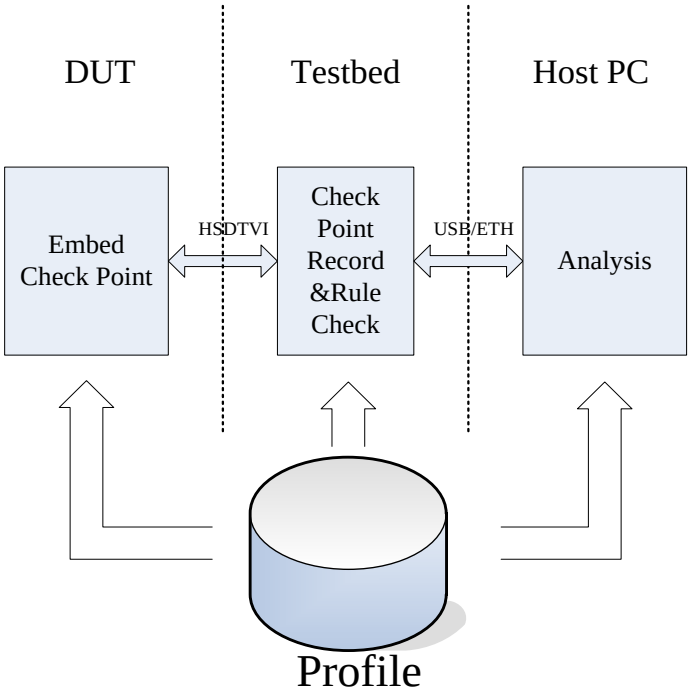

Figure 4-9 The HSDTVI Debug Trace and Validate Process

The HSDTVI Slave is Design under Test (DUT). The HSDTVI Master is a testbed. DUT will continually send checkpoints to testbed. Those checkpoints will reflect the running state of DUT. Testbed then records checkpoints in real-time. Each record of checkpoints for a period will form a profile for this given period. Through analysis of these profiles, DUT's state can be decoded. Therefore, these profiles can greatly help to detect and locate the bug in DUT.

Because the DataBus is an 8-bit parallel GPIO Port, so the HSDTVI Interface can send an 8-bit checkpoint status with only three instructions. The ultra-low overhead eases the placement of checkpoints, so they can put in IRQ handler without affecting the performance of system.

In order to make full use of the HSDTVI interface, the software on iLive needs embedded checkpoints into important running stage, such as starting/stopping sensing sensor, starting/stopping transferring RF data, receiving a RF packet, receiving an external event, etc. Then iLive can send these checkpoints to Raspberry Pi Boards through the HSDTVI in real-time. Due to the light overhead of the HSDTVI operation, these checkpoints can be put in anywhere in the program (system or application), even in IRQ handler.

These detailed and precise checkpoints log will form a profile of iLive related to a specific period. Based on the profile, the run path and state of iLive can be easily decoded. With necessary tool for analyzing and comparing profile, the HSDTVI can help user to build a useful automated debug test and validate environment.

The software on Raspberry Pi Board can catch and store the checkpoints from iLive. Beyond the checkpoint, the timestamp of checkpoint is also very important. Thanks to the 1MHz hardware system timer in Raspberry Pi, the timestamp can be accurate to one micro second period.

The CPU of Raspberry Pi Board is a 700 MHz ARM1176JZF-S core (Broadcom.com, 2013), comparing with 8 MHz 8-bit AVR microcontroller in iLive, the Raspberry Pi Board is over hundreds times more performance than iLive. The computation resource is enough for the tracing and logging checkpoints from iLive.

4.4.3.2. Real-time Fault Detection Mode

Figure 4-10 shows the HSDTVI interface used for real-time fault detection mode scenario. In this scenario, the HSDTVI Slave is an 8-bit AVR/AVRRF microcontroller with IEEE802.15.4 wireless access media; the HSDTVI Master is only a low power NanoRisc, whose power consumption is much lower than HSDTVI Slave, only 1 percent of AVR. This low power consumption NanoRisc can greatly help to improve not only the reliability, but also the lifetime.

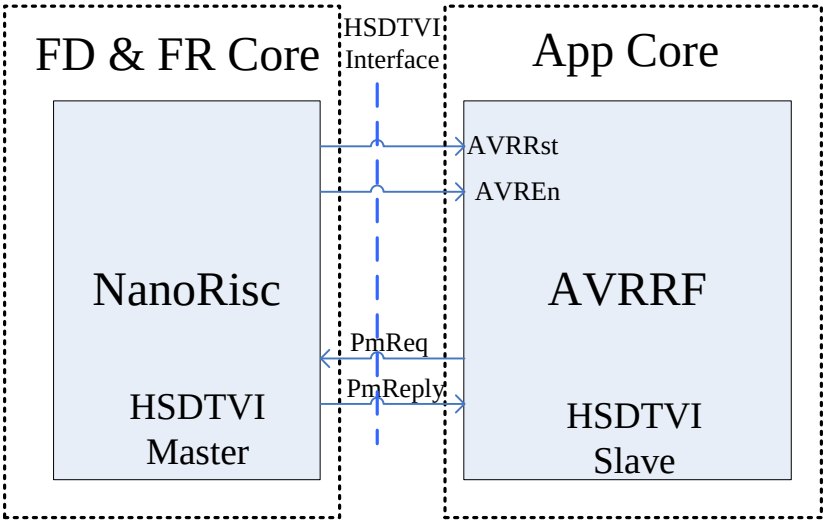

Figure 4-10 The HSDTVI Interface used for real-time Fault Detection Mode Scenario

Table 4-3 details related Pin between NanoRisc and AVRRF used for real-time fault detection mode.

Table 4-3 The HSDVTI Pin connections between NanoRisc and AVR

Pin Name	Direction	Pin Description	Related Functions	
			On AVRRF	On NanoRisc
PmReq	AVRRF➔NanoRisc	AVRRF informs NanoRisc that it has finished its job (request to power down, for end-device node), High voltage is active	SetPmReqOn SetPmReqOff	PmReqIsOn PmReqIsOff
PmReply	AVRRF➔NanoRisc	NanoRisc provide ACK to AVRRF, High voltage is active	PmRespIsOn PmRespIsOff	SetPmReplyOn SetPmReplyOff
AVRRst	AVRRF➔NanoRisc	NanoRisc use it to reset AVRRF, longer than 300ns low voltage pulse can reset AVRRF	N/A	SendAVRRst
AVREn	NanoRisc➔PSU	NanoRisc use this pin to control the power supply of AVRRF, High is active the Power Source for AVRRF	N/A	SetAVREnOn SetAVREnOff

Due to different node types, the node functions differently. Therefore, the HSDTVI communication protocol also needs to change a little bit to meet the different requirements. Here we mainly discuss two main different node types: Coordinator and End-device.

In fact as coordinator, the AVR is always wakeup, so the AVR will use PmReq as heart beat signal. AVR will send one PmReq pulse every circle. NanoRisc will reset or power on/off AVR when the PmReq pulse has not occurred in time or the PmReq pulse is too longer.

Figure 4-11 shows the timing diagram of normal heart beat check of coordinator.

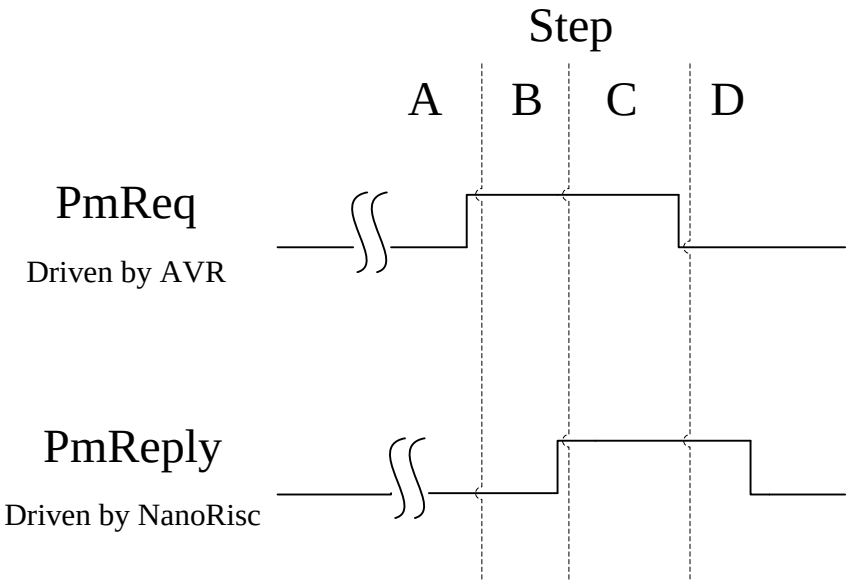

Figure 4-11 Timing diagram of AVR and NanoRisc Communication

Table 4-4 and Table 4-5 show the pseudo code for this heartbeat checking process in the coordinator.

Table 4-4 Pseudo Code for Heart Beat Checking of Coordinator

AVR	Direction	NanoRisc	Comment

//Active PmReq
SetPmReqOn();
	→	//Wait PmReqIsOn
		While(PmReqIsOff()
		&& !Timeout());
			Step A

//Wait PmReplyIsOn
While(PmReplyIsOff()
&& !Timeout());
	←	//Active PmReply
		If(PmReqIsOn())
		SetPmReplyOn();
		Else
		goto Err;
			Step B

//Deactive PmReq
If(PmReplyIsOn)
	→	//Wait PmReqIsOff
		While(PmReqIsOn()
			Step C

AVR	Direction	NanoRisc	Comment
SetPmReqOff() Else Goto Err;		&& !Timeout());	

//Wait PmReplyIsOff
While(PmReplyIsOn()
&& !Timeout());
If(PmReplyIsOn())
Goto Err;
	←	//DeActive PmReply
		If(PmReqIsOff)
		SetPmReplyOff();
		Else
		goto Err;
			Step D

Table 4-5 Pseudo Code for Error Handle of the coordinator

AVR		NanoRisc
//Report Err Type to Local Server Err: ReportErr();		//Power Off Node & Reboot Node Err: SetAVREnOff(); Delay(2000ms); SetAVREnOn(); SendAVRRst();

On End-device, sleep and wakeup mode is adopted. In fact to minize energy consuming most of the time, the AVR of the End-device is powered off. AVR will be powered on only when needed. This sleep and wakeup mechanical can greatly improve the lifetime of WSN end-device node. In order to inform its work status to NanoRisc, AVR will send one PmReq pulse after it finished sensing and sending job. If NanoRisc receives PmReq, it will power off AVR gracefully. Otherwise, it may directly power off AVR without confirmation from AVR.

Table 4-6 and Table 4-7 shows the pseudo code for this heartbeat checking process.

Table 4-6 Pseudo Code for Heart Beat Checking of End-device

AVR	Direction	NanoRisc

//Active PmReq
SetPmReqOn();
	→	//Wait PmReqIsOn
		While(PmReqIsOff() && !Timeout());

//Wait PmReplyIsOn
While(PmReplyIsOff() && !Timeout());
	←	//Active PmReply
		If(PmReqIsOn())
		SetPmReplyOn();

AVR	Direction	NanoRisc
		Else goto Err;

//Save Data & Deactive PmReq
If(PmReplyIsOn)
{
SaveInfoBeforePowerOff();
SetPmReqOff()
}
Else
Goto Err;
	→	//Wait PmReqIsOff
		While(PmReqIsOn() && !Timeout());

//Wait PmReplyIsOff
While(PmReplyIsOn() && !Timeout());
If(PmReplyIsOn())
Goto Err;
	←	//DeActive PmReply
		If(PmReqIsOff)
		{
		SetPmReplyOff();
		SetAVREnOff();
		}
		Else
		goto Err;

Table 4-7 Pseudo Code for Error Handle of End-device

AVR		NanoRisc
//Record Err Type Err: RecordErr();		//Power Off Node Err: SetAVREnOff();

4.4.4. Summary

The HSDTVI Interface provides a new basic method to debug, test and validate the microcontroller running state in real-time. The main features of the HSDTVI Interface include:

- Light overhead: A checkpoint needs only three instructions. Sending 8-bit checkpoint need less than 1μs using AVR while with the same function using UART (38400bps) needs 260μs

- Easy to use, can debug interrupt handler and concurrent programs: The checkpoint related codes can be placed at anywhere in the program including in interrupt handler
- Can help to localize the dysfunction of an application (real-time fault checking)
- Ease the detection fail and recovery
- Real-time debug trace & verify
- Real world debug trace & verify: the HSDTVI Master can be deployed in real world environment embedded into the HSDTVI Slave. In this case, it helps user to locate bugs show up only in physical environment
- Provide the key technology as Auto-Tester: the HSDTVI Master can run suitable software to check real-time state of the HSDTVI Slave. The software can act according to the result of fault detection on the HSDTVI Slave. Therefore, the HSDTVI Master can help developer to check the program automatically, easy for regression tests or long time monitor for transient error
- Force design-for-test way: Request developers to provide the profile of check points, the check rules in profile will be used by the HSDTVI Master. This potentially help to ensure the whole develop process following the design-for-test way
- Detect failure more quickly and more accurately: Checkpoint can be put in anyplace in the program, and it also can be designed with the inside logic of SW, these extra information in checkpoints can help to detect HW/SW failure more quickly and accurately
- Fault injection support: Fault injections are necessary to test, validate and evaluate the reliability of a system to short the test and validation time. The HSDTVI interface can help to gather the results of fault injection.
- Support mutual debug, test, fault detection and fault recovery. The HSDTVI is a bidirectional communication bus. So it can help to implement mutual real-time debug, test, fault detection and fault recovery.

4.5. Summary

In this chapter, we discussed multicore WSN node architecture and the special HSDTVI interface in the new architecture. The multicore WSN node architecture enables the development and implementation of new dependable and energy efficiency wireless sensor network.

Chapter 5. High Reliability Design Process dedicated to Resource Constraint Embedded System

5.1. Introduction

The efficient and robust realization of the non-conventional multicore wireless sensor network is a challenging algorithmic and technological task. The multicore architecture is more complex than the single core architecture. Key features including high resource constraints, high reliability requirements, various sensor types, dynamical wireless environments, and huge numbers of WSN nodes in different autonomous group force us to change every aspects of our design process.

Based on many years of real world project experiences, we propose a new design process HRDP (High Reliability Design Process dedicated to High Resource Constraint Embedded System) to guide our development. In HRDP, we will implement an integrated multicore platform (WSN node, Hardware support, fault injection testbed) supporting run time testing and validation. Furthermore, we will use fault injection technique to help to discovery the reason of soft errors. Through the discovery and understood those soft errors and recovery from failure, HRDP can greatly help to improve the overall system. We hope the new integrated HRDP can allow both to simplify the testing and validation (hardware and software) and to improve the reliability of WSN node. The rest part of this chapter will detail the content of new design process.

5.2. Traditional Design Process Models

Many design process models have been developed in order to achieve different required objectives. We briefly discuss some frequently employed models, e.g. Waterfall Model, V Model, Incremental Model, Spiral Model Model-Driven Engineering, RAD Model and Agile Model.

5.2.1. Waterfall Model

The Waterfall Model was the first design process model to be introduced. It is also referred to as a linear-sequential life cycle model. The Waterfall Model is the most rigid one, suggesting to move to a phase only when its preceding phase is completed and perfected. Phases of development in the waterfall model are kept completely separated, and there is no room for iteration or overlap (Benington, 1983). Figure 5-1 shows the diagram of the Waterfall Model.

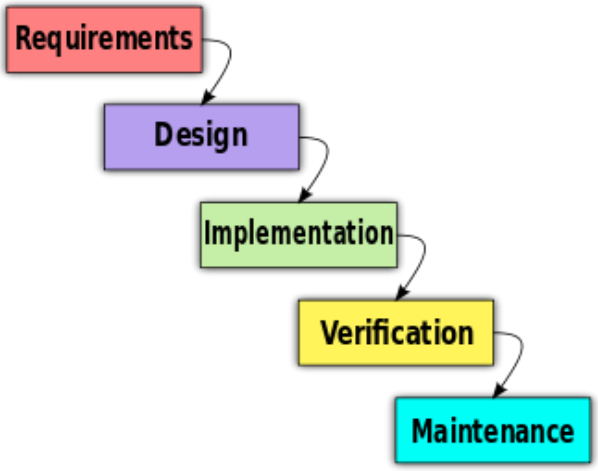

Figure 5-1 Block diagram of Waterfall Model

Waterfall Model is very simple and easy to understand and use. However, it is only capable to model simple, clear, well known and fix requirements project. Due to the rigid one-way rules, it will have high amounts of risk and uncertainty in the late stage. Therefore, Waterfall Model is not a good model for complex and long projects (A. C. S. Beck et al., 2012).

5.2.2. V Model

The V Model has the same strict serial structure as the waterfall model, but it suggests that, before going to a more detailed design level, one should already test all the system features and properties that can be tested at the current level of design abstraction (BRUMMOND, CONGER, HART, OSBORNE, & ZAREAN, 2006). Figure 5-2 shows the diagram of the V Model.

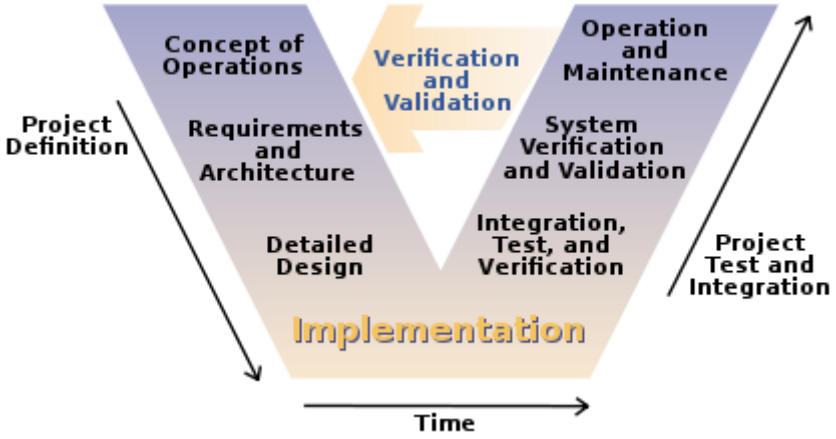

Figure 5-2 The V-model of the Systems Engineering Process

V Model is simple and easy to use. The test activities happen before implementation, so the defects can be found in the early stage. This can greatly help to avoid the downward flow of defects. It is still very rigid and least flexible. The system is developed during the implementation phase, so no early simulation or prototypes of the system are produced. Therefore, V Model is only good for small projects, which requirements need to be easily understood (Nowka, 2007).

5.2.3. Incremental Model

The Incremental Model divides the whole requirement into various builds. Each build passes through the requirements, design, implementation and testing phases. Each subsequent release of the build adds function to the previous release. The process continues until the complete system is achieved. Multiple development cycles make the Incremental Model a multi-waterfall process (Larman & Basili, 2003; Pressman, 2010). Figure 5-3 shows the diagram of the Incremental Model.

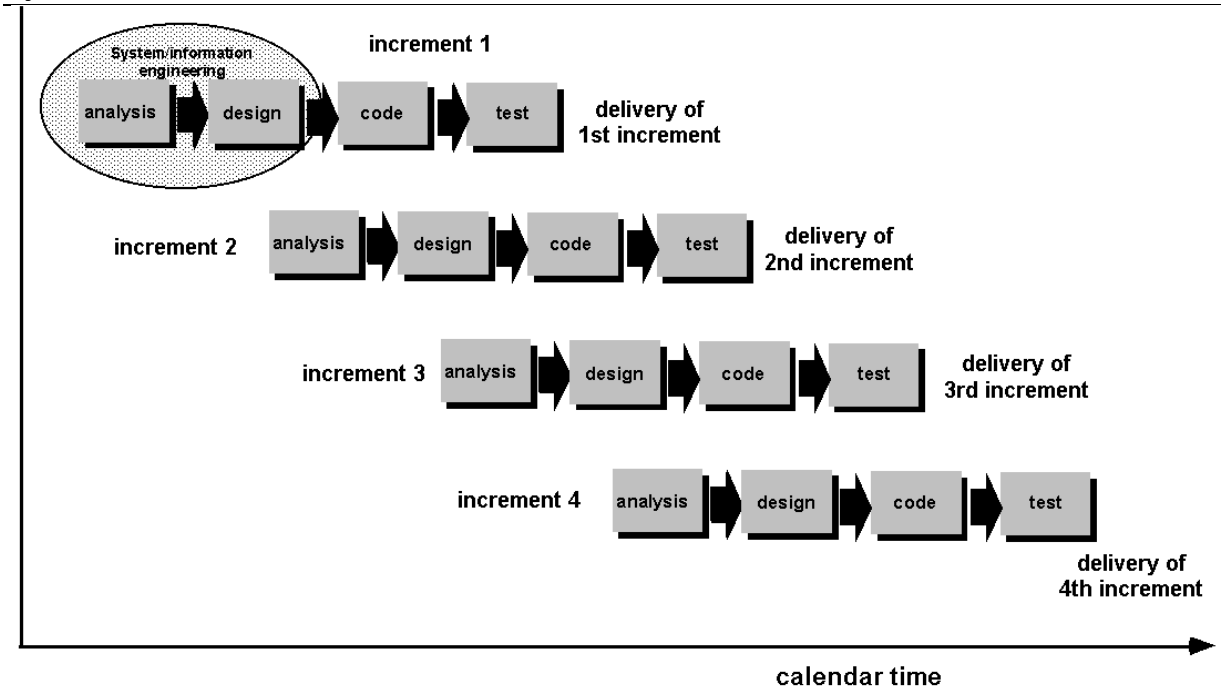

Figure 5-3 The Incremental Model of Development

Incremental Model can generate prototype quickly and early. It brings more flexible and lower initial delivery cost. In addition, the increments can greatly help to manage the risk. However, the Incremental Model requires a clear and complete definition of whole system before it can be broken down and build incremental. Moreover, the total cost is higher due to the multi increments (Nowka, 2007).

5.2.4. Spiral Model

The Spiral Model is similar to the incremental model, with more emphases placed on risk analysis. The Spiral is visualized as a design process passing through some number of iterations, with the four-quadrant diagram representative of the following activities:

- Formulate plans to: identify software targets, implement the program, clarify the project development restrictions
- Risk analysis: an analytical assessment of selected programs, to consider how to identify and eliminate risk
- Implementation of the project: the implementation of software development and verification

The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model) (Boehm, 1986). Figure 5-4 shows the diagram of the Spiral Model.

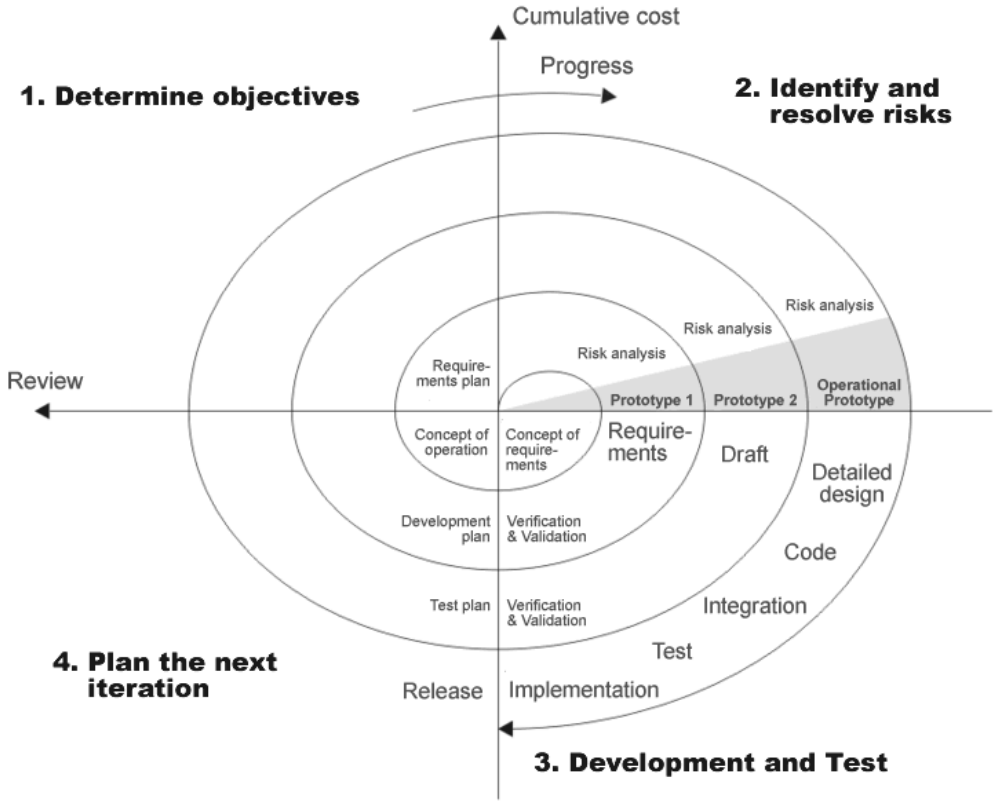

Figure 5-4 The Spiral model of the Systems Engineering Process

The Spiral Model pays more attention on the risk analysis. It has strong approval and documentation control and early produced software to help to avoid risk. It is good for large and mission-critical projects. However, it can be a costly model, and requires highly specific expertise on risk analysis. It does not work well for smaller projects (Nowka, 2007).

5.2.5. RAD Model

Rapid Application Development (RAD) model is a type of incremental model. In RAD model, the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype. RAD model uses modeling concepts to capture information about business, data, and processes. This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements (Martin, 1991). Figure 5-5 shows the diagram of the RAD Model.

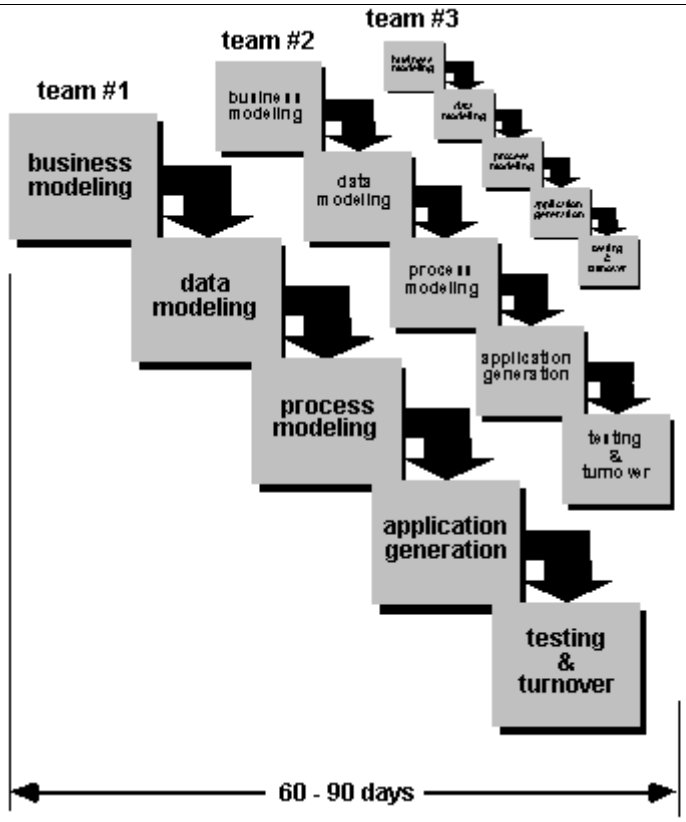

Figure 5-5 The Rapid Application Development (RAD) Model

The RAD Model can reduce the development time, increase reusability of components, provide quick initial reviews, encourages customer feedback, and integrate from very beginning solves a lot of integration issues. However, RAD requires highly skilled developers/designers and the cost of modeling and automated code generation is very high (Nowka, 2007).

5.2.6. Agile Model

Agile development model is also a type of Incremental model. Software is developed in incremental and rapid cycles. This results in small incremental releases with each release is built on previous functionality. Each release is thoroughly tested to ensure software quality is maintained. It is used for time critical applications. Extreme Programming (XP) is currently one of the most well-known agile development life cycle model (K. Beck et al., 2001; Prolinx Services, 2013). Figure 5-6 shows the diagram of the Agile Model.

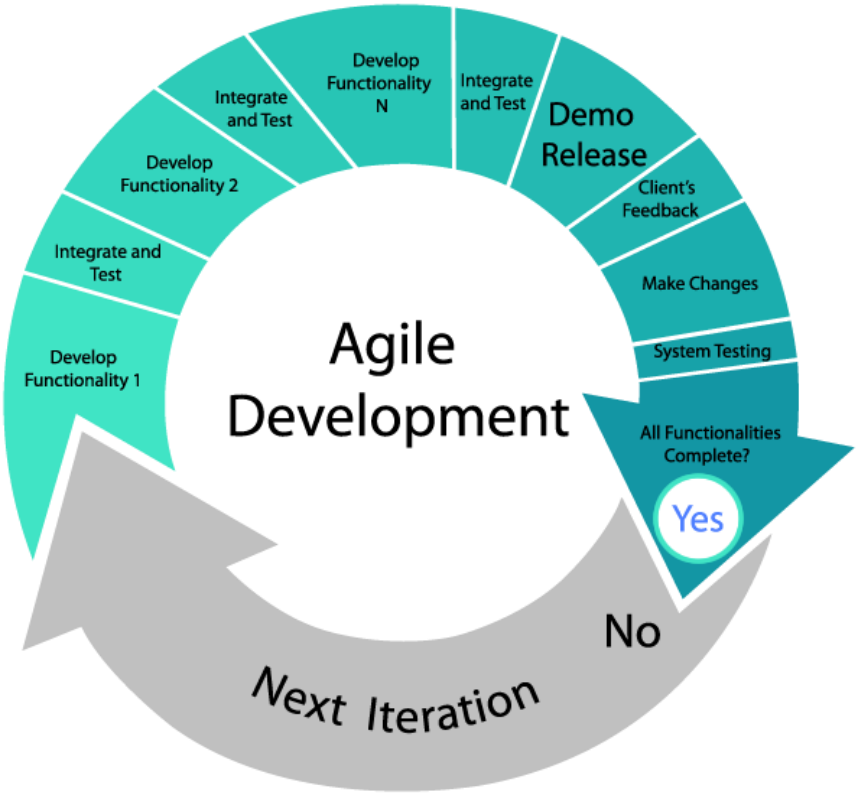

Figure 5-6 The Agile Development Model

Agile development model can provide rapid, continuous delivery of useful software for customer. In Agile development model, people and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other. Agile development model can accept late changes in requirements. Continuous attention helps to create technical excellence and good design. However, Agile development model lack of emphasis on necessary designing and documentation. Only senior programmers are capable of taking the kind of decisions required during the development process (Nowka, 2007).

5.2.7. Summary

Due to the rigid and least flexible rules, Waterfall Model and V Model is not good model for the development of our multicore wireless sensor network. The cost of Incremental Model or Spiral Model is too high. The RAD Model and Agile development model are mainly focus on the software development. Therefore, we propose a new design process, High Reliability Design Process dedicated to High Resource Constraint Embedded System (HRDP), in the implementation of multicore WSN node.

5.3. High Reliability Design Process Based on Multicore Architecture

The design process can be viewed as a sequence of steps that transforms a set of specifications described informally into a detailed specification that can be used for manufacturing. All the intermediate steps are characterized by a transformation from a more abstract description to a more detailed one.

In this part, we propose a new design process named as High Reliability Design Process dedicated to High Resource Constraint Embedded System (HRDP) based on multicore architecture. It tries to ease the development of multicore WSN node and improve the productivity and system quality.

5.3.1. General Overview

The HRDP is a design process for a multicore WSN node. Therefore, it is assumed that the top architecture of the node should be multicore architecture. It is also assumed that the HSDTVI interface will be implemented.

Normally a project has four important Product Life Cycle (PLC) phases:

- Conception Phase: Collect product requirements
- Design Phase: Architecture design, implementation of hardware, software and mechanical design, as well as test
- Realization Phase: Manufacture
- Service Phase: Installation, Operation & Maintain WSN

The following part will detail the HRDP in design phase, including architecture design, early validation and test.

5.3.2. Model-Driven Engineering

Model-driven engineering (MDE) is a software development methodology which focuses on creating and exploiting domain models (that is, abstract representations of the knowledge and activities that govern a particular application domain), rather than on the computing (or algorithmic) concepts (Frankel, 2003; Haan, 2009).

The MDE approach is meant to increase productivity by maximizing compatibility between systems (via reuse of standardized models), simplifying the process of design (via

models of recurring design patterns in the application domain), and promoting communication between individuals and teams working on the system (via a standardization of the terminology and the best practices used in the application domain).

A modeling paradigm for MDE is considered effective if its models make sense from the point of view of a user that is familiar with the domain, and if they can serve as a basis for implementing systems. The models are developed through extensive communication among product managers, designers, developers and users of the application domain. As the models approach completion, they enable the development of software and systems.

Some of the better known MDE initiatives are:

- the Object Management Group (OMG) initiative model-driven architecture (MDA), which is a registered trademark of OMG.
- the Eclipse ecosystem of programming and modeling tools (Eclipse Modeling Framework).

Figure 5-7 shows the diagram of the MDE.

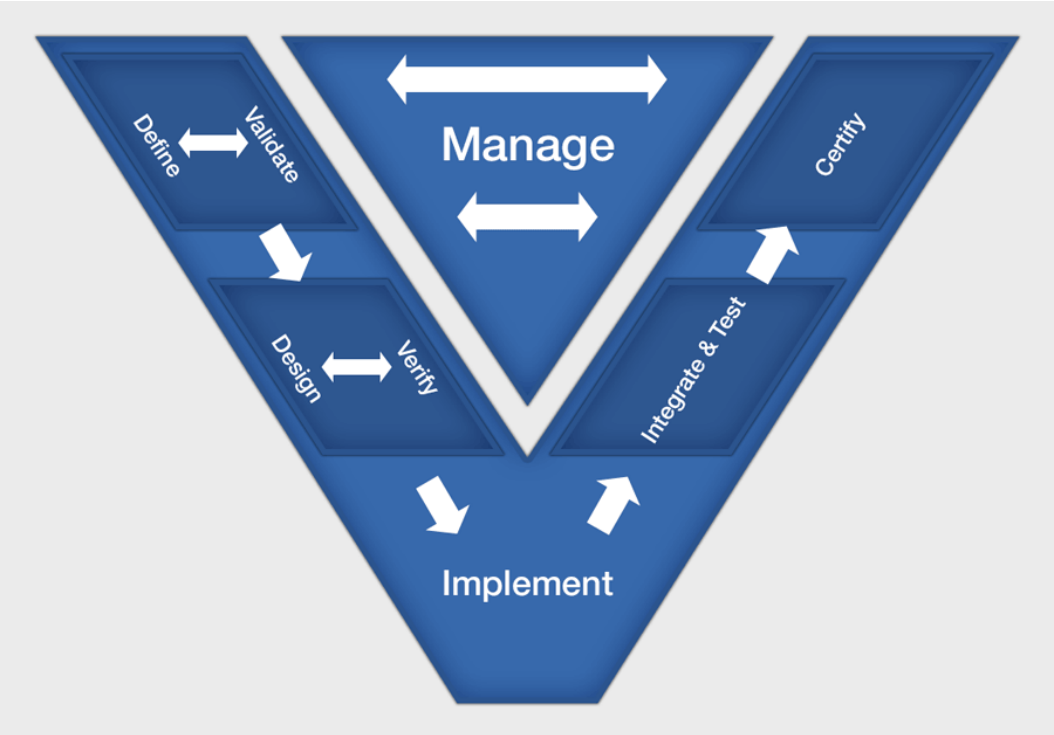

Figure 5-7 Overview of Model Driven Engineering

The HRDP is a specific MDE dedicated to multicore architecture. The HRDP follows the MDE concept, which is an improved V-Model by supporting the test phases at each design level by software models that simulate the system before real implementations exist already.

5.3.3. Model of Multicore WSN Architecture

5.3.3.1. Top Level Architecture

Top Level Architecture provides one or more diagrams depicting an overview of the target solution architecture with supporting narrative text. Ensure that the diagram(s) depict the major components of the solution and the relationships between the components, input and output data flows, major processes, functions, and system tasks. Identify major Commercial-Off-the-Shelf (COTS), infrastructure, and platform technology components.

Figure 5-8 below is an example of top-level multicore WSN architecture diagram.

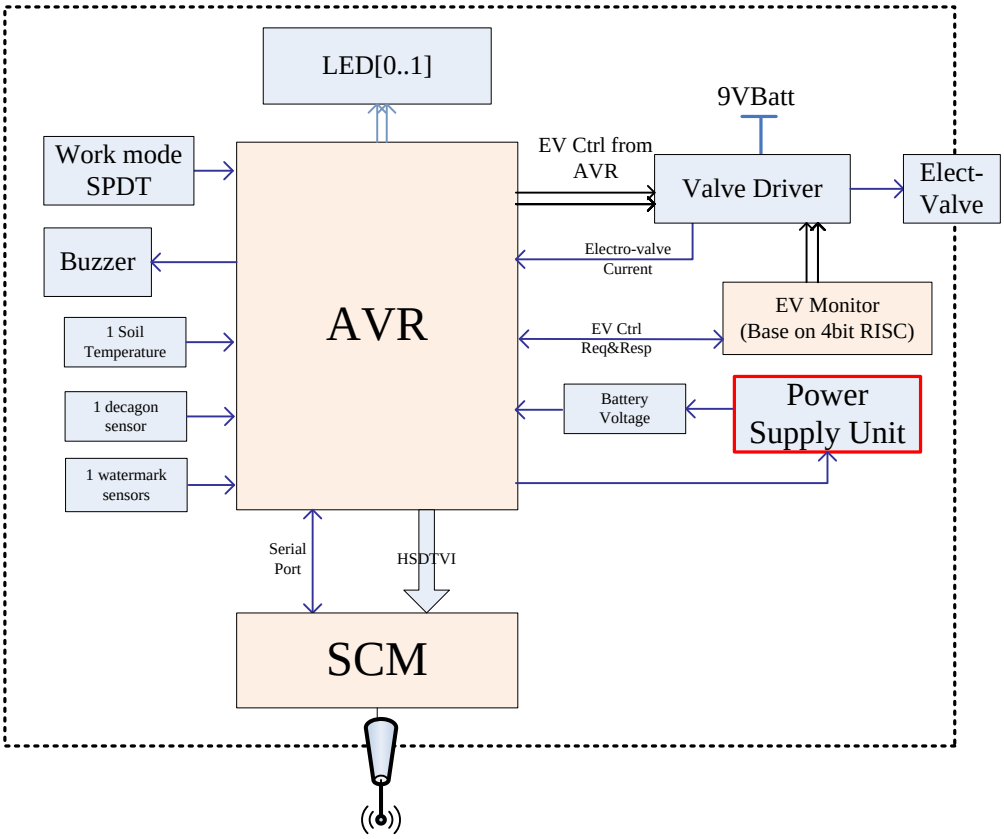

Figure 5-8 Example Multicore WSN Architecture Diagram

5.3.3.2. Modules List

The Table 5-1 provides the modules list in the Figure 5-8.

Table 5-1 Example Module List

Module Name	Module Description
cAVR	AVR Micro-controller of SIS
cSCM	SCM in SIS, support wireless access to SIS, also connects cAVR with the HSDTVI, can support debug trace and validate software in cAVR.
cEvDrv	Driver of Electrovalve
cEvMonRisc	Electrovalve Monitor 4-bit RISC
cSisMod	Work Mode Switch of SIS
cPSU	Power Supply Unit of SIS
cSensor	Soil Temperature and Soil Moisture Sensor
cLed	1 Green Led and 1 Red Led
cBuzzer	1 Buzzer

5.3.3.3. Module Interface Specification

Module Interface specification identifies all interfaces between high-level modules and other modules or systems. Following are examples of module interface table for two different modules.

5.3.3.3.1. Interface of Module with Software inside

Table 5-2 Example High-level Interface of cAVR Module

Pin Name	Direction	Pin Description	Related Functions	
			Function Name	Comment
SisMod0	Input	From Work Mode Switch, Low mean SIS is on Auto work mode	SisMode0IsOn SisMode0IsOff	Low voltage is active
SisMod1	Input	From Work Mode Switch, Low mean SIS is on Manual work mode	SisMode1IsOn SisMode1IsOff	Low voltage is active
AVRRst	Input	Low For Hardware Reset AVR	N/A	
AVREvCtlPlus	Output	To EV Driver, Low Request Electrovalve Open	SetAVREvCtlPlusOn SetAVREvCtlPlusOff	Low voltage is active
AVREvCtlMinus	Output	To EV Driver, Low Request Electrovalve Close	SetAVREvCtlMinusOn SetAVREvCtlMinusOff	Low voltage is active
EvCur	Input	Analogy Input from EV Driver, Indicate the current of EV	ReadEvCurData	
EvCtlReq	Output	To cEvMonRisc, Low Request Electrovalve start work	SetEvCtlReqOn SetEvCtlReqOff	Low voltage is active

Pin Name	Direction	Pin Description	Related Functions	
			Function Name	Comment
EvCtlReply	Input	From cEvMonRisc, Low for Response	EvCtlReplyIsOn EvCtlReplyIsOff	Low voltage is active
GLed	Output	Green Led	SetGLedOn SetGLedOff	Low voltage is active
RLed	Output	Red Led	SetRLedOn SetRLedOff	Low voltage is active
BZ	Output	Buzzer	SetBZOn SetBZOff	Active is 2Khz PWM
SenEn	Output	Sensor Power Enable Signal	SetSenEnOn SetSenEnOff	AVR already have
SoilTemp	Input	1-Wire Soil Temperature Sensor	ReadSoilTempData	
DeSoil	Input	Decagon Soil Moisture Sensor	ReadDeData	
WmSoil	Input	Watermark Soil Moisture Sensor	ReadWmData	
BatV	Input	Battery Voltage	ReadBatVData	
Hsdtvi	Output	Hardware support debug trace and validate interface, output to cSCM	WriteHsdtviData	
Uart	I/O	Serial communication bus with cSCM	ReadUartData WriteUartData	

5.3.3.3.2. Interface of Module without Software inside

Table 5-3 Example High-level Interface of cEvDrv Module

Pin Name	Direction	Pin Description
SCMEvCtlPlus	Input	From cSCM, Low Request Forward Pulse for Electrovalve
SCMEvCtlMinus	Input	From cSCM, Low Request Reverse Pulse for Electrovalve
EmEvCtlPlus	Input	From cEvMonRisc, Low Request Forward Pulse for Electrovalve
EmEvCtlMinus	Input	From cEvMonRisc, Low Request Reverse Pulse for Electrovalve
EvPulse	Output	To Electrovalve, Forward Pulse will Open Electrovalve, Reverse Pulse will Close Electrovalve, otherwise Electrovalve will remain current Open/Close status.
EvCur	Output	Analogy Output, Indicate the current of EV

5.3.3.4. Functional Specification of Each Module

Functional Specification identifies the functionality of all high-level modules. For complex module, the functional specification can use C/C++ language or other high-level language to describe.

Following Table 5-4 is an example of a simple module functional specification.

Table 5-4 Sample Functional Specification of cEvDrv Module

SCMEvCtlPlus	EmEvCtlPlus	SCMEvCtlMinus	EmEvCtlMinus	EvPulse	Comment
On	On	Off	Off	Forward Pulse	Remain 50ms, Electrovalve will open
Off	Off	On	On	Reverse Pulse	Remain 50ms, Electrovalve will Close
Off	Off	On	Off	Reverse Pulse	Remain 50ms, Electrovalve will Close
Off	Off	Off	On	Reverse Pulse	Remain 50ms, Electrovalve will Close
Off	Off	Off	Off	No Output	Electrovalve will remain current status
On	Off	Off	Off	No Output	Electrovalve will remain current status
Off	On	Off	Off	No Output	Electrovalve will remain current status
On	On	On	Off	Not Allowed	Waste power
On	On	Off	On	Not Allowed	Waste power
On	On	On	On	Not Allowed	Waste power

5.3.4. Early Validation of Requirement

The HRDP supports three types of early validation of requirement:

- Validate by designer

- Validate based on MDE
- Validate based on virtual processor emulator

The following part will detail these three types of early validation.

5.3.4.1. Validate by Designer

The simplest way to validate the requirement is directly checked and validated by the design team. This method requires highly skilled developers/designers. Only senior designers are capable of taking the kind of decisions required during the development process.

Therefore, the method is only good for redesign projects, which requirement has only slightly modification.

5.3.4.2. Validate based on MDE

With suitable MDE Tool, such as AADL (Feiler & Gluch, 2012), UML (Lavagno, Martin, & Selic, 2003), SysML (Holt, Perry, Engineering, & Technology, 2008), etc.. Notice that our early architecture can directly run in Model-Driven Development Environment (MDDE).

Even though this method requires studying and learning MDE and MDDE, the formal verification and automatic synthesis of implementations with MDE can greatly help to guarantee robust and safety of our design.

The biggest shortage of MDE is that the cost of development tool. The free tool may lack many mature models, while the commercial tool's price is still expensive.

Figure 5-9 shows a block diagram for the early validation based on AADL. My colleague, ZHOU Peng, in our SMIR team focuses on this direction.

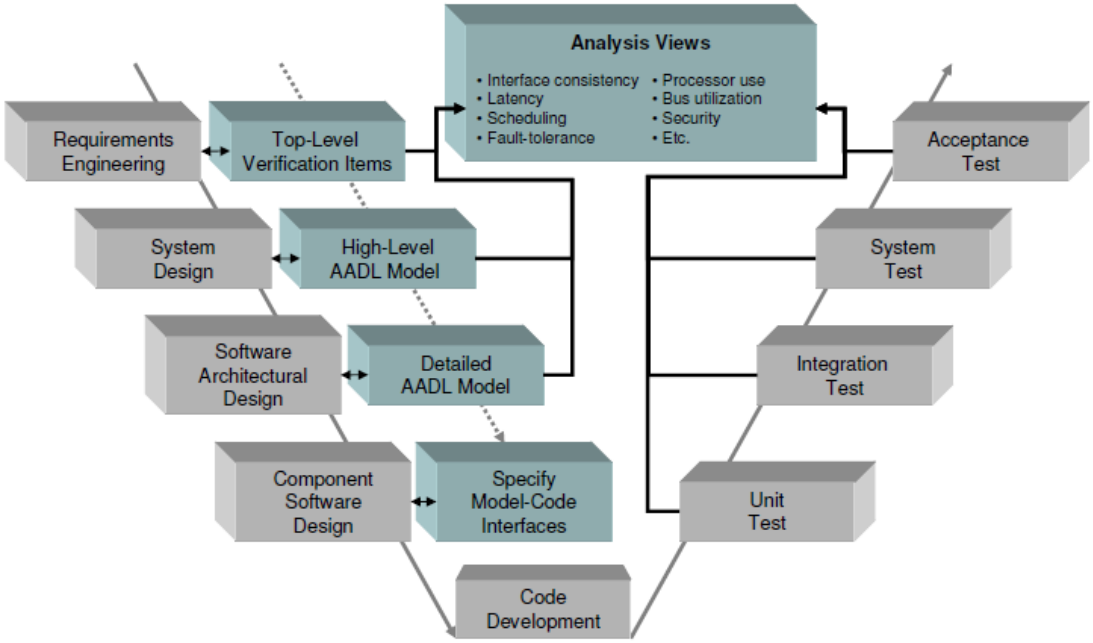

Figure 5-9 Early Validation Based on AADL

5.3.4.3. Validate based on Virtual Processor Emulator

In this method, virtual processor emulator such as Cooja (Osterlind, 2006), QEMU(Bellard, 2005), can be adopted to provide a virtual platform for simulating the software.

The virtual processor emulator can run the software directly, so the validate process only requires necessary adding new virtual hardware driver or modification on exist virtual hardware driver for new requirements.

This method can formally verify the design without extra cost on MDE tool, such as buying, studying or learning. Besides, most of the source code can directly reuse in later design stage.

Figure 5-10 shows a block diagram for the early validation based on virtual processor emulator. My colleague, de VAULX Christophe, in our SMIR team focuses on this direction.

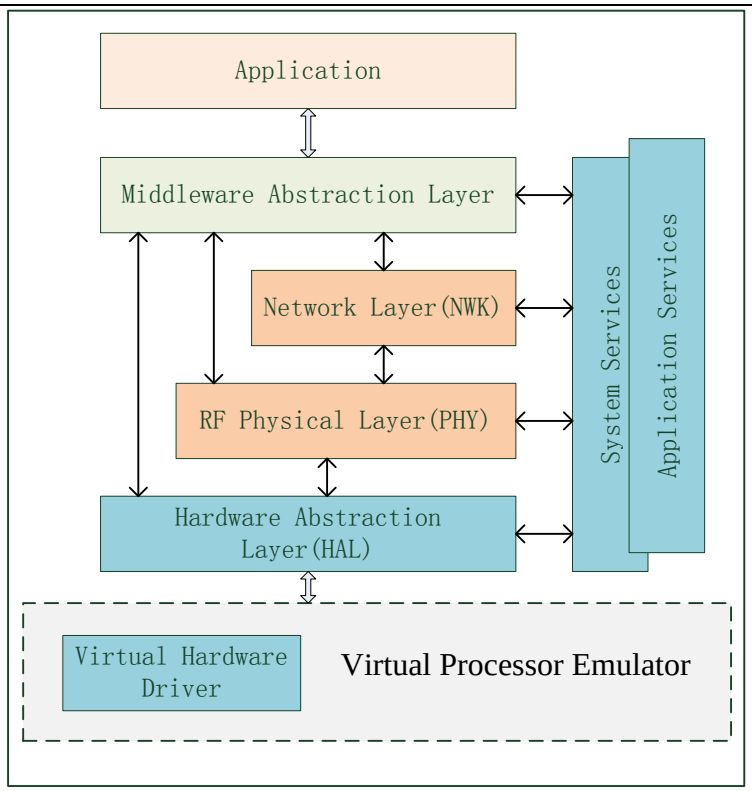

Figure 5-10 Early Validation Based on Virtual Processor Emulator

5.3.5. Design for Multicore Run Time Testability

Every design, hardware and software, should be tested. Kent Beck said, "Code that isn't tested doesn't work - this seems to be the safe assumption." However, not all designs are easy to test. More often than not, the effort invested into testing a specific area is in inverse proportion to evaluate how it can be tested easily. Put simply, the easier parts of the system to test, get tested a lot more than those that are harder to test. Testing is a major activity in any development lifecycle - a large part of a project budget is spent on it. If we want to use effectively it, user friendly testing environment should be addressed from the early stages of system design.

While in multicore WSN architecture, the HSDTVI interface provides a hardware supported to ease the testability not only in debug period, but also in real-world run time period.

The Design for Multicore Run Time Testability (DMRTT) related to several parts:

- Can help to Detect and localize bug more Quickly and more Accurately
- Support Real-Time Fault Detection, help to improve the reliability of the system
- Force Design-For-Testability Way: The fault detection code in test bench of each module is requested to run in the HSDTVI master. This potentially help to ensure the

whole develop process following the Design-For-Testability Way. When the HSDTVI Master real-time using these rules to detect the state of HSDTVI Slave, the method is working like DMRTT.

5.3.6. Fault Injection

As we mentioned before, in real world deployment, 10%-20% of single core WSN nodes left the network in the first week with unknown reason. With the mutual real time checking and validation between cores based on multi cores and HSDTVI, the multicore WSN node can automatically recover from this type fault. However, we still do not understand the real cause of the fault. That why we propose to use fault injection method into HRDP to discover the reason of the fault, understand more about the fault and late help to improve the overall system.

Figure 5-11 shows a block diagram for the fault injection testbed. Testbed mainly consist four parts: Design Under Test (DUT), Fault Injection Board, PC and peripheral equipment.

- The DUT is the WSN node under test.
- The Fault Injection Board is a specification board for fault injection to WSN node based on Raspberry Pi board.
- The PC controls the test and records/displays the results.
- The peripheral equipment includes off-the-shelf peripheral equipment Atmel AVR JTAG MK-II and SuperPro USB Programmer to program the microcontroller on DUT, EMI Burst-Generator and EMI-Probe, Californium-252 Source for Heavy-Ion Radiation. The pre-validated WSN node is special peripheral equipment, which running IEEE802.15.4 stack. Therefore, the DUT can test its RF link with this node without expensive RF Wave Generator and Analyzer.

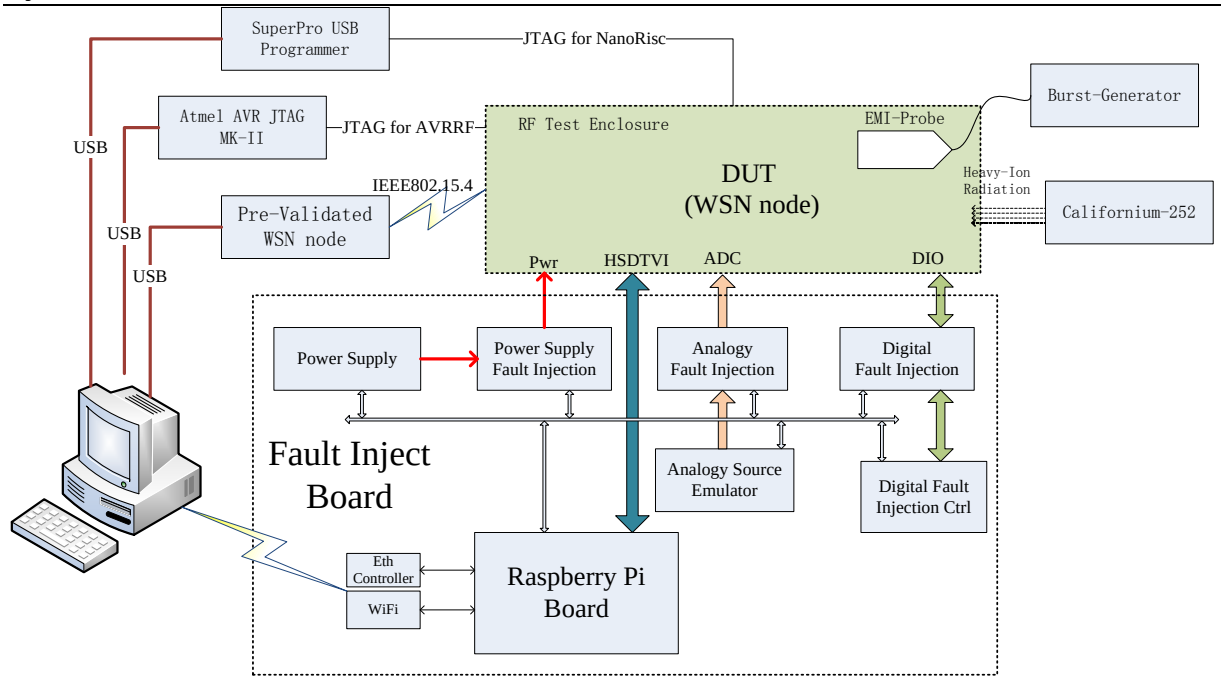

Figure 5-11 Block diagram of the Fault Injection TestBed

The Figure 5-12 shows the circuit board of the Fault Injection TestBed.

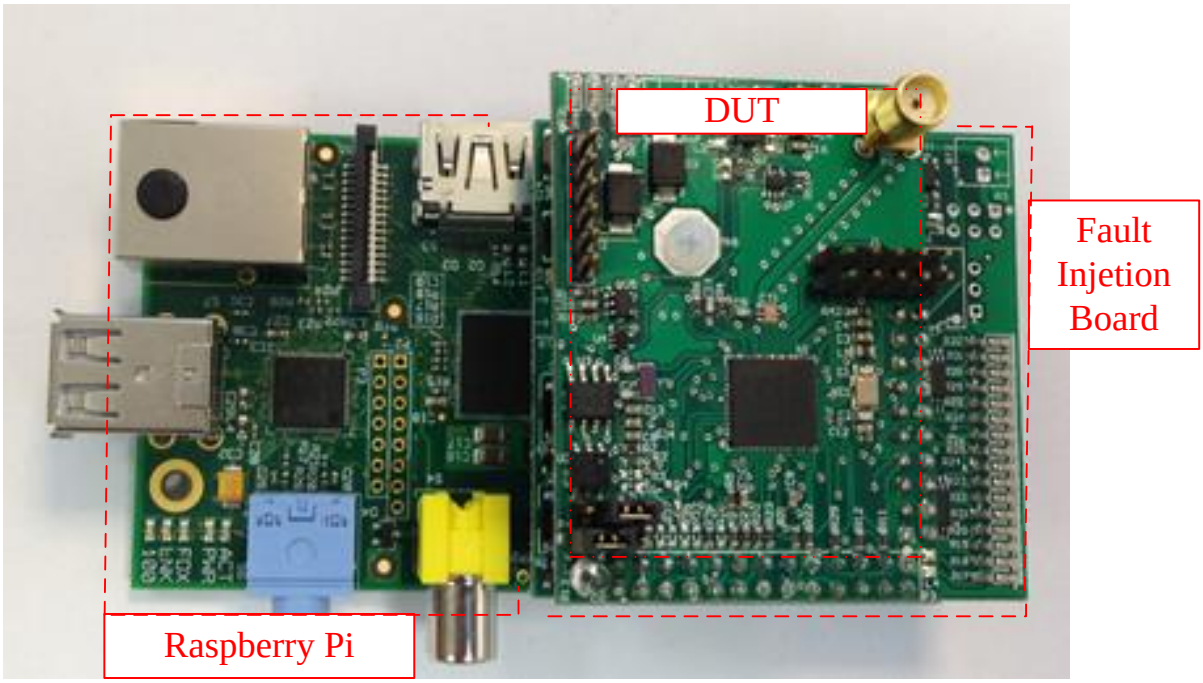

Figure 5-12 Circuit Board of the Fault Injection TestBed

The main features of Fault Injection Board are:

- Support Inject fault in Power supply of DUT
- Support Inject fault in Analog ADC port of DUT

- Support Inject fault in Digital I/O port of DUT
- Sample the power supply voltage and current of DUT.
- Independent Power supply (different from DUT board),
- HSDTVI port: Debug and trace interface, Raspberry Pi board can use this port to trace and log all the checkpoints sending from DUT. According to the checkpoints, RaspberryPi board can RESET or switch off the DUT.
- Features based on Raspberry Pi
  - 700 MHz ARM1176JZF-S core (ARM11 family)
  - Up to 512MB SDRAM
  - Ethernet interface.
  - WiFi module supported.
  - Support SD Card for Large data Storage.

The Table 5-5 provides the hardware fault injection with contact supported by Fault Injection Board.

Table 5-5 Fault Injection modes with contact

Signal Type	Fault Injection Method	Parameters			
		Period	Duration	Voltage	Freq
Power Supply	Under shoot	Adj	Adj	Adj	
	Over shoot	Adj	Adj	Adj	
	Stuck-at GND	Adj	Adj		
	Stuck-at Voltage	Adj	Adj	Adj	
	Open	Adj	Adj		
	Noisy	Adj	Adj	Adj	Adj
Analogy Signal	Stuck-at GND	Adj	Adj		
	Stuck-at Voltage	Adj	Adj	Adj	
	Open	Adj	Adj		
	Noisy	Adj	Adj		
Digital Signal	Stuck-at GND	Adj	Adj		
	Stuck-at VDD	Adj	Adj		
	Stuck-at H	Adj	Adj		
	Stuck-at L	Adj	Adj		
	Open	Adj	Adj		
	Flip	Adj	Adj		

The Table 5-6 provides the hardware fault injection without contact supported by Test Equipment, EMI Burst-Generator and Californium-252 Source, and Pre-validated WSN node.

Table 5-6 Fault Injection modes without contact

Signal Type	Fault Injection Method	Parameters				
		Period	Duration	Voltage	Freq	Fluence
Electro Magnetic Interference	Burst (Transients)	Adj	Adj	Adj	Adj	
	ESD	Adj	Adj	Adj		
	Surges	Adj	Adj	Adj		
	PQT (Voltage Dips)	Adj	Adj	Adj		
Radiation	Heavy-Ion Radiation	Adj	Adj			Adj
RF	RF Interference	Adj	Adj		Adj	

Of course, developer can also inject any software fault through the JTAG debugger. Therefore, through the testbed in Figure 5-11, developer can injection any kind of fault, hardware or software, to speed up the debug test and validate process. And the testbed will also ease the development of the multicore software for it can greatly help to discover unknown fault.

5.4. Summary

Adopting the HRDP methodology allows systems architects, software engineers, and hardware designers to achieve the following objectives:

- The HRDP can provide an early prototype for the validation of requirement
- The HRDP enables higher abstract layer designing and verification
- With suitable tool support, the HRDP enables auto translation from design to production implementation
- The models in the HRDP increase the reusability of engineering resource
- The models also can improve the flexibility of deployment (and redeployment) of engineering resources
- The HRDP help to Detect and localize bug more Quickly and more Accurately
- The fault injection testbed can help to discover unknown fault and late improve the overall system

All this benefit can help to improve productivity and system quality.

Chapter 6. Implementation of Multicore WSN Node

We develop several hardware platforms to validate multicore WSN architecture, which is technology and application independent. We name these platforms as E²MWSN, iLive, SIS, iLiveEdge, EPER, RPiER, etc. These hardware platforms are instances of multicore WSN architecture with varying degrees of hardware complexity. We will analyze each specific implementation in the following sections.

6.1. E²MWSN: High Reliability and High Performance Multicore WSN Node

6.1.1. General Overview

The E²MWSN is a complex instance of a multicore WSN architecture. The objective of the E²MWSN is to implement a configurable and energy efficient multicore WSN node for outdoor/indoor applications. Thanks to the configurability of multicore, the E²MWSN is able to adapt to diverse applications domains, from simple data collecting application to complex real-time control application.

Figure 6-1 shows the block diagram of E²MWSN. E²MWSN is built on three cores architecture. The Main App Core in E²MWSN is a low power 8-bit RISC ATMEGA1281. The Auxiliary Core in E²MWSN is a low power ARM7TDMI 32-bit RISC AT91SAM7Sx. We choose an ultra-low power FPGA IGLOO to be the FD & FR Core. The Auxiliary Core AT91SAM7Sx is more powerful than the Main App Core. So if the WSN application has complex computational processes such as signal processing, data compressing, encryption, decryption etc., AT91SAM7Sx can be activated to handle these computation when necessary. The FD & FR Core IGLOO is the control center of the E²MWSN. It controls power supply sources for all the cores and devices. The FD & FR Core also run as a monitor of Main App Core. If it detected faults in the Main App Core, it will isolate the faulty Main App Core and active the Auxiliary Core to substitute the Main App Core. Through the switching of core, the E²MWSN can provide seamless services even in the presence of faults.

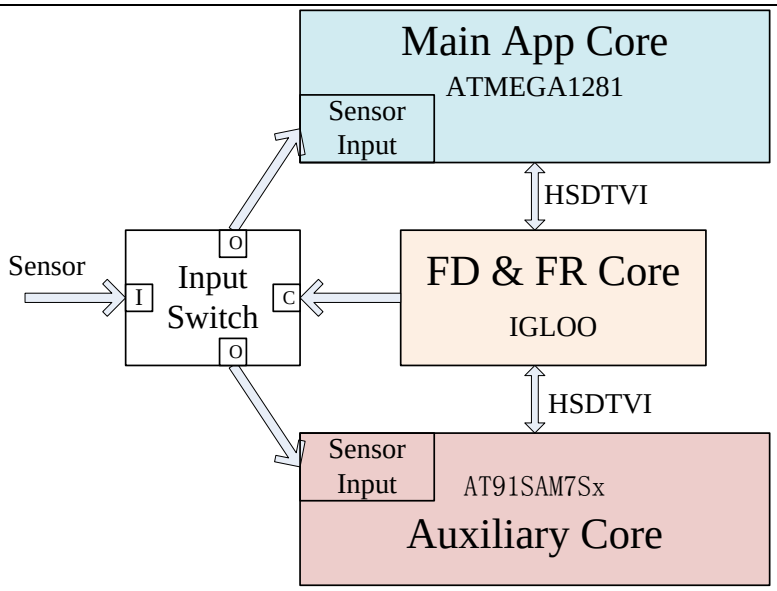

Figure 6-1 Block diagram of the E³MWSN

6.1.2. Hardware Architecture

Figure 6-2 presents the hardware architecture of E³MWSN. These three cores in the E³MWSN share the I²C, UART and SPI bus. Each core can be activated to run as the master of those communication buses. The AT91SAM7Sx and the ATMEGA1281 share the analog ports, one of or both cores can turn on the ADC to sample the values from analog sensor such as Decagon soil moisture sensor and battery voltage sensor. The circuit can independently control the power supply of each core and device.

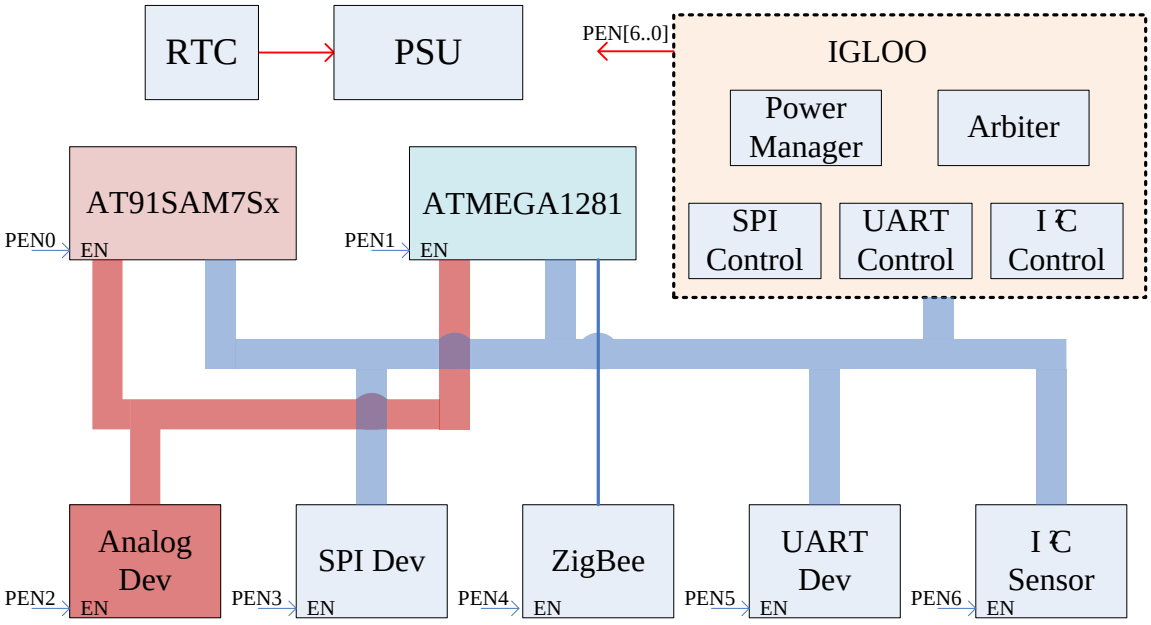

Figure 6-2 Hardware Architecture of E³MWSN

The most powerful Core in the E²MWSNis the AT91SAM7Sx running at 48 MHz and delivering about forty-three Million Instructions Per Second (MIPS) (Atmel-Corporation, 2011). This 32-bit RISC has: 512-Kbyte flash program memory, 64-Kbyte static RAM, internal 8-channel 10-bit analog-to-digital converter, three hardware timers, thirty-two general-purpose I/O lines, one USB 2.0 full speed (12 Mbps) device port, two external Universal Synchronous/Asynchronous Receiver Transmitter (USART), one I²C interface and one master/slave Serial Peripheral Interface (SPI) port.

The ATMEGA1281 is running at 8 MHz and delivering about eight MIPS (Atmel-Corporation, 2012b). This 8-bit microcontroller has 128-Kbyte flash program memory, 8-Kbyte static RAM, internal 8-channel 10-bit analog-to-digital converter, 3 hardware timers, 48 general-purpose I/O lines, 1 external Universal Asynchronous Receiver Transmitter (UART), and one SPI port.

The IGLOO mainly works as the configurable network connector for the devices on board. With the configurable IGLOO, the circuit can change the connections between cores; adjust work states of all cores without making any wired change. The IGLOO may be configured to implement specific processing (image or signal processing e.g.) when need. The IGLOO family of flash FPGAs, based on a 130-nm flash process, offers the lowest power FPGA, a single-chip solution, small footprint packages, reprogram ability, and an abundance of advanced features (Actel-Corporation, 2009). The Flash*Freeze technology used in IGLOO devices enables entering and exiting an ultra-low-power mode that consumes nano Power while retaining SRAM and register data. Flash*Freeze technology simplifies power management through I/O and clock management with rapid recovery to operation mode. The Low Power Active capability (static idle) allows for ultra-low-power consumption while the IGLOO device is completely functional in the system. This allows the IGLOO device to control system power management based on external inputs (e.g., scanning for Passive Infrared Motion Detector output stimulus) while consuming minimal power.

Figure 6-3 shows the implemented board of the E²MWSN.

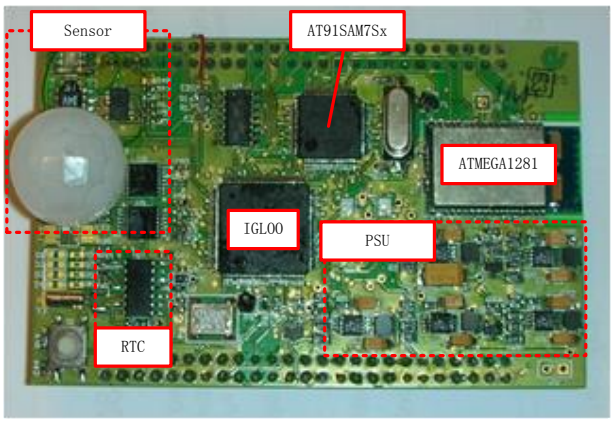

Figure 6-3 Circuit Board of the E²MWSN

6.1.3. Key Features

The major features of the E³MWSN are listed below:

- Ultra-low power consumption
- Dimension 85mm*54mm
- 1 light sensor
- 1 temperature sensor
- 1 air humidity sensor
- 1 Passive Infrared Motion Detector
- 3 Decagon soil moisture sensors
- 2 RS232 ports
- 1 USB slave port
- 1 Real-time Clock (RTC)
- SD Card
- IEEE802.15.4 ZigBee wireless access medium

6.1.4. Performance

6.1.4.1. Power Consumption

The circuit can controls independently the power supply of each core, so the E³MWSN can work in several modes as presented in Table 6-1.

Table 6-1 Operation modes of the E³MWSN

Operation Modes	Status of each Core		
	AT91SAM7Sx	**ATMEGA1281**	**IGLOO**
single AT91SAM7Sx	ON	OFF	OFF
single ATMEGA1281	OFF	ON	OFF
single IGLOO	OFF	OFF	ON
AT91SAM7Sx plus ATMEGA1281	ON	ON	OFF
IGLOO plus ATMEGA1281	OFF	ON	ON
IGLOO plus AT91SAM7Sx	ON	OFF	ON
AT91SAM7Sx plus ATMEGA1281 plus IGLOO	ON	ON	ON

Operation Modes	Status of each Core		
	AT91SAM7Sx	ATMEGA1281	IGLOO
Deep sleep(RTC ON)	OFF	OFF	OFF

Here we mainly compare single AT91SAM7Sx mode (ATMEGA1281 is closed), single ATMEGA1281 mode (AT91SAM7Sx is closed), and AT91SAM7Sx plus ATMEGA1281 mode.

To evaluate the power consumption of the E²MWSN on different operation modes, the total running current of the E²MWSN is measured as Figure 6-4.

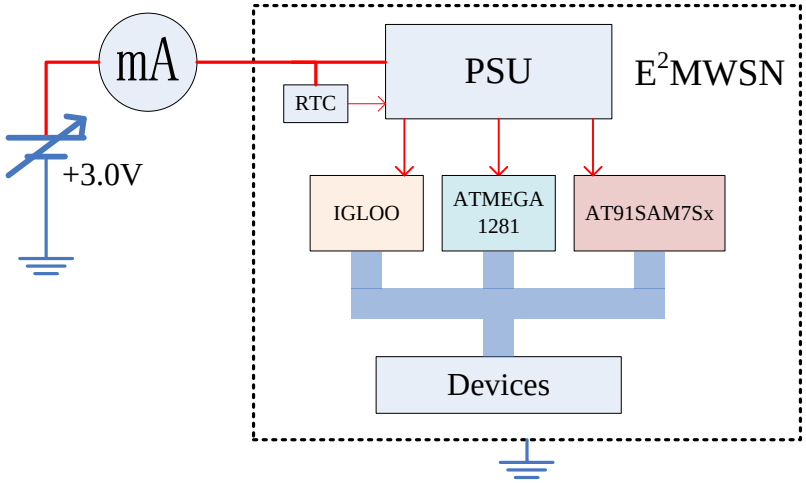

Figure 6-4 Measure Schematics of the E²MWSN

A DC power supply ELC AL936 provided a +3.0V output as emulator of two AA Batteries. A Metrix MX53 in mA gear position measured the current of the E²MWSN when the node handles various tasks including sensing, signal processing, data storage, and wireless communication on the three modes mentioned previously. The results are recorded in the Table 6-2, Table 6-3, and Table 6-4 as follow.

Table 6-2 Task Resource Required of the E²MWSN on single AT91SAM7Sx mode

Task	Resource Required		Energy Consumption(μJ)
	Current	Time Used	
Sensing	22.1mA	896ms	59405
Signal Processing	19.7mA	5.0ms	295
Data Storage	20.9mA	6.0ms	376
Wireless Communication*	21.8mA	140ms	9156
Sleep	0.2mA	178953ms**	107372
Total			176604

Table 6-3 Task Resource Required of the E²MWSN on single ATMEGA1281mode

Task	Resource Required		Energy Consumption(μJ)
	Current	Time Used	
Sensing	15.9mA	900ms	42795
Signal Processing	9.9mA	268ms	7919
Data Storage	16.3mA	10.1ms	494
Wireless Communication	21.8mA	140ms	9156
Sleep	40μA	178682ms**	21442
Total			81806

Table 6-4 Task Resource Required on AT91SAM7Sx plus ATMEGA1281 mode

Task	Resource Required		Energy Consumption(μJ)
	Current	Time Used	
Sensing	15.9mA	900ms	42795
Signal Processing	19.7mA	5.0ms	295
Data Storage	20.9mA	6.0ms	376
Wireless Communication	21.8mA	140ms	9156
Sleep	1μA	178949ms**	537
Total			53159

*ATMEGA1281 is used for wireless access.

**Sleep time is determined by sample frequency. In this dissertation the sample frequency is 3 minutes per sample, which is the same as the TelosB one (J. Polastre et al., 2004).

The total power consumption of each mode can be calculated by (4.3) and (4.4), we can get:

$$\xi\left(A_{AT91SAM7Sx}\right)=176604\,\mu J\ , \tag{6.1}$$

$$\xi\left(A_{ATMEGA1281}\right)=81806\,\mu J\ , \tag{6.2}$$

$$\xi^{M}\left(A_{AT91SAM7Sx+ATMEGA1281}\right)=53159\mu J\ . \tag{6.3}$$

The result is $\xi\left(A_{AT91SAM7Sx}\right)>\xi\left(A_{ATMEGA1281}\right)>\xi^{M}\left(A_{AT91SAM7Sx+ATMEGA1281}\right)$, being accordance with the estimation in (4.5).

Since the E²MWSN mulitcore mode is more energy efficient, it can achieve longer lifetimes than other mode. With a pair of AA Lithium/Iron Disulfide (Li/FeS_2) 3000mAh batteries and the sampling period is 3 minutes, the lifetime of the E²MWSN mulitcore mode is 1270 days. For comparison, the lifetime of single AT91SAM7Sx mode is 382 days, single ATMEGA1281 mode is 825 days, and the TelosB is 945 days.

6.1.4.2. Reliability

The Mean Time between Failure (MTBF) and Mean Time between Critical Failure (MTBCF) is calculated to evaluate the reliability of the E²MWSN. The methods used to analyze the MTBF/MTBCF are MIL-HDBK-217 (US, 1997) and FIDES 2009 (DGA, September 2010). MIL-HDBK-217 is considered to be the most common used reliability prediction method, but it has not been revised since 1995 (issue F notice 2). Compare with MIL-HDBK-217, FIDES is a newer reliability assessment method. The FIDES can take into consideration new technologies, so the FIDES is more accurate with the new components, such as Commercial off-the-shelf (COTS) components. Therefore, if manufacturer provides the FIT data of component, we will use it first. Otherwise, if the component is COTS such as microcontroller, we will use FIDES method. Finally, the rest parts will be analyzed by MIL-HDBK-217 method.

Here, each core and related external components such as power supply unit, oscillator, decoupling capacitor etc. are combined together to calculate. The Failures in Time Failure Rate in Parts per Billion Hours (FIT) of each core are listed as following:

Table 6-5 FIT of Each Core in the E²MWSN*

Core	FIT
ATMEGA1281	48.07
AT91SAM7Sx	58.93
IGLOO	22.39

*The raw MTBF or FIT data is taken from manufacturers (Atmel-Corporation, 2012c; Kemet, 2012; Linear, 2009; Microsemi-Corporation, 2011; Onsemi, 2012).

Because single ATMEGA1281 mode and single AT91SAM7Sx mode do not have redundancy, so the MTBF and MTBCF of these two modes are the same. However, in AT91SAM7Sx plus ATMEGA1281mode, AT91SAM7Sx and ATMEGA1281 is parallel backup of each other. So the MTBCF will much bigger than MTBF in this mode. The MTBF/MTBCF of three work modes is listed as following:

Table 6-6 MTBF/MTBCF of Each E²MWSN Operation Mode

Operation Mode	Overall MTBF/MTBCF	
	MTBF	MTBCF
single ATMEGA1281	1.42E+07	1.42E+07
single AT91SAM7Sx	1.23E+07	1.23E+07
AT91SAM7Sx plus ATMEGA1281	7.73E+06	4.47E+07

From Table 6-6, we can know that the AT91SAM7Sx plus ATMEGA1281 mode is the highest reliable mode among these three works modes. The MTBCF of this mode is 4.47E+07 hour, over 5000 years.

As mentioned in Figure 2-4 and 2.2.1.6, the MTBF/MTBCF is the failure rate (bottom of the bathtub curve) of core components, and it is not the MTBF/MTBCF of the WSN node board. It is also not the product lifetime of WSN node.

6.2. iLive: High Reliability and Low cost Multicore WSN Node

6.2.1. General Overview

ILive is a simple and cheap instance of multicore WSN architecture. The objective of iLive is to implement a low cost and high reliable multicore WSN node for environment data collection and precision agriculture applications. It is a dissymmetric two cores architecture. The Auxiliary Core is removed. Figure 6-5 shows the block diagram of iLive.

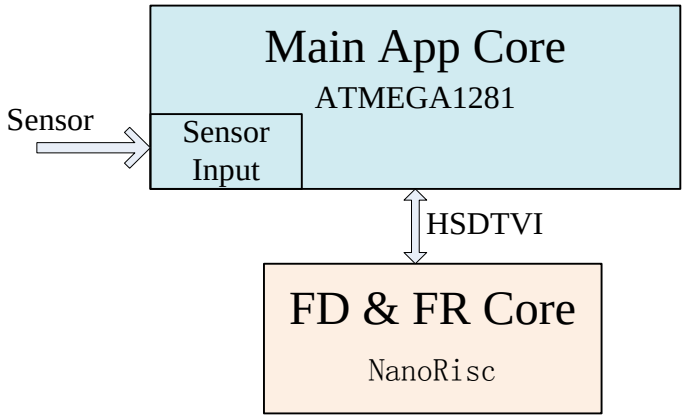

Figure 6-5 Block diagram of iLive

The Main App Core in iLive is a low power 8-bit RISC ATMEGA1281. The FD & FR Core in iLive is an ultra-low power 4-bit RISC NanoRisc. The FD & FR Core runs as a

monitor of ATMEGA1281. If it detected faults in the ATMEGA1281, it can generate reset signal for ATMEGA1281 or directly power off ATMEGA1281 when necessary to fix most of faults. Moreover, the NanoRisc consumes only one percent of energy of Main App Core, so it can great help iLive to archive longer lifetime when iLive work in Sleep &Wakeup mode.

6.2.2. Hardware Architecture

The Figure 6-6 presents the hardware architecture of iLive. In order to ease the development of environment data collection and precision agriculture application, iLive has two types of soil moisture probes: watermark sensor and decagon sensor. It can directly support four Watermark sensors and three Decagon sensors without adapter board. It integrates on board one air temperature sensor, one air humidity sensor and one light sensor based on I²C bus. It has a UART port, which can directly support USB-to-Serial converter cable to connect with a PC.

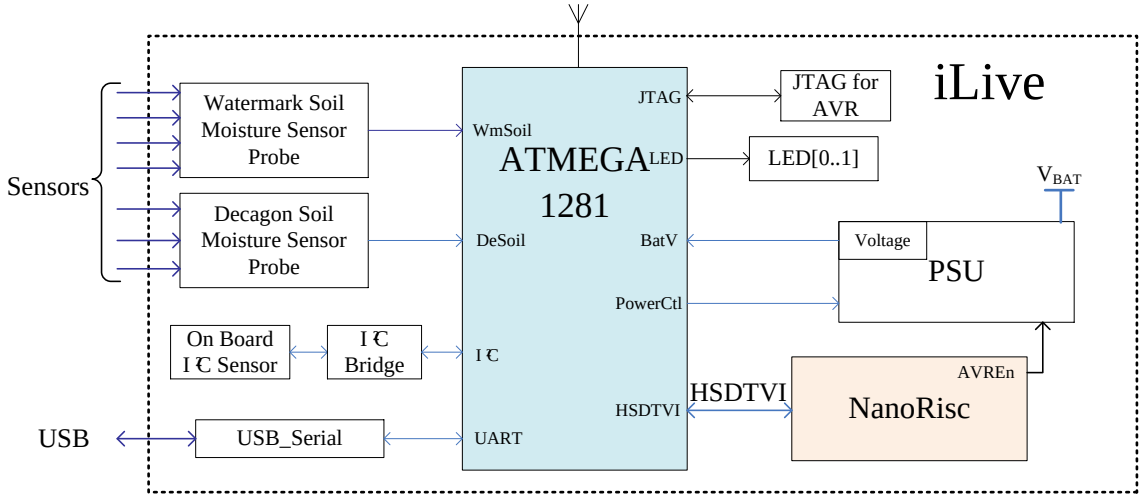

Figure 6-6 Hardware Architecture of the iLive

The ATMEGA1281 core in iLive is the same one in the E²MWSN, which has 128-Kbyte of flash program memory, 8-Kbyte of static RAM and running at 8MHz. The NanoRisc is an ultra-low power 4-bit microcontroller coming in a small 8-pin SO package and working up to 0.4 MIPS. It consumes only 5.8 µA in active mode and 3.3 µA in standby mode. On the contrary, ATMEGA1281 need 500 µA in active mode and 130 µA in standby mode (Atmel-Corporation, 2012b). Base on the ultra-low power feature of NanoRisc, it can greatly help to improve the lifetime of iLive when iLive work in Sleep &Wakeup mode. Moreover, it requires no external component, so it is very easy to be integrated in iLive. The NanoRisc contains the equivalent of 8 kB of Flash memory and a RC oscillator, which is configurable to oscillate from 32 to 800 kHz. It also has an integrated 4-bit ADC, a power-on reset, watchdog timer, 10-bit up/down counter, PWM and several clock functions. It has a sleep counter reset allowing automatic wake-up from sleep mode. It is designed for use in battery-operated and

field-powered applications requiring an extended lifetime. A high integration level makes it an ideal choice for cost sensitive applications.

Figure 6-7shows the implemented board of iLive.

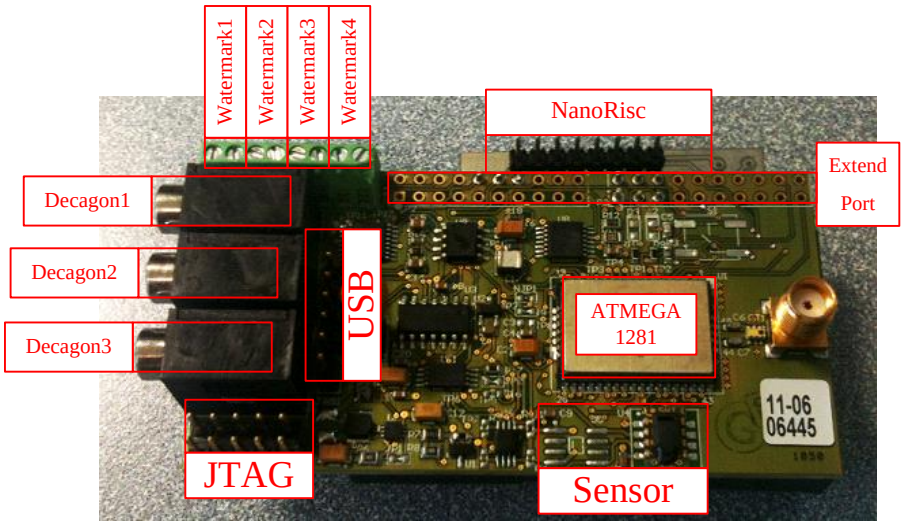

Figure 6-7 Circuit Board of iLive

6.2.3. End-Device Sleep & Wakeup Work Mode

The End-Device iLive node has three running modes: Deep sleep mode, Sleep mode and Active mode. In Active mode, the AVR RISC of iLive will gather the sensor data and transfer data to coordinator through IEEE802.15.4 wireless access media. In Sleep mode, the AVR RISC will power off sensors and RF components, and remain in sleep mode to decrease the power consumption. In deep sleep mode, the AVR RISC will also be powered off to further decrease the total power consumption of iLive. Thess features enable iLive achieve a very long battery lifetime.

Figure 6-8 shows the timing diagram of iLive. Every time iLive be connected with power supply source, it will continuously power on AVR for T_{Init}^{ON} . During this period, the iLive will switch only between Active mode (remain for T_{Active}) and Sleep mode (remain for T_{Sleep}). Therefore, user can use JTAG or USB-to-Serial Convert cable to upgrade the firmware or middleware of iLive without meeting power fail problem.

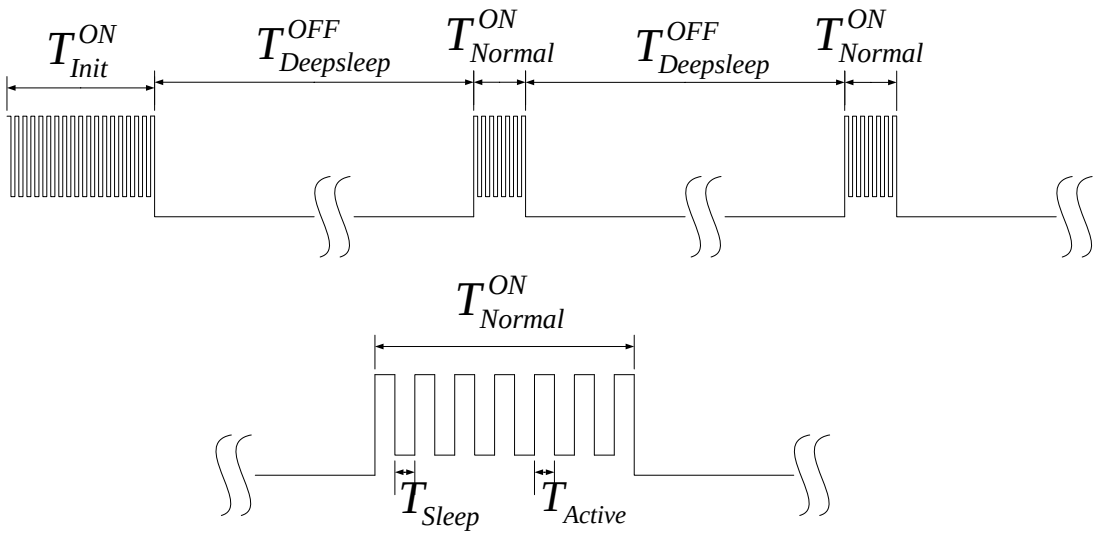

Figure 6-8 Timing Diagram of iLive

After T_{Init}^{ON}, iLive will switch to deep sleep mode for $T_{Deepsleep}^{OFF}$ to save the energy. Because the AVR is powered off, so iLive will not response to any event, such as JTAG, Serial port, RF, etc.

When iLive will be powered on AVR again for T_{Normal}^{ON}. The T_{Normal}^{ON} running mode of iLive is similar as in T_{Init}^{ON}, iLive will switch between Active mode (remain for T_{Active}) and Sleep mode (remain for T_{Sleep}). The $T_{Deepsleep}^{OFF}$ and T_{Normal}^{ON} form the Sleep & Wakeup loop of iLive.

Table 6-7 provides the default timing parameters of iLive.

Table 6-7 Default Timing Parameters of iLive

Parameter Name	Value	Modification	Comment
T_{Init}^{ON}	15 min	Fixed	
$T_{Deepsleep}^{OFF}$	1 Hour	Fixed	
T_{Normal}^{ON}	1 min	Fixed	
T_{Sleep}	5 sec	Define by Middleware	
T_{Active}	3 sec	Related to the Sensor type and count	

6.2.4. Key Features

The main features of iLive are following:

- Ultra-low energy consumption, 2 AA standard batteries for 5 years*;
- Dimension 76mm*40mm
- 4 Watermark sensors
- 3 Decagon sensors
- 1 temperature sensor
- 1 air humidity sensor
- 1 light sensor
- 1 RS232/USB slave port
- IEEE802.15.4 ZigBee wireless access medium.

* 1 sample per day with 4 watermark sensors

6.2.5. Performance

6.2.5.1. Power Consumption

To evaluate the power consumption of the iLive, the total running current of iLive is measured as shown by in the Figure 6-9, similar to the Figure 6-4.

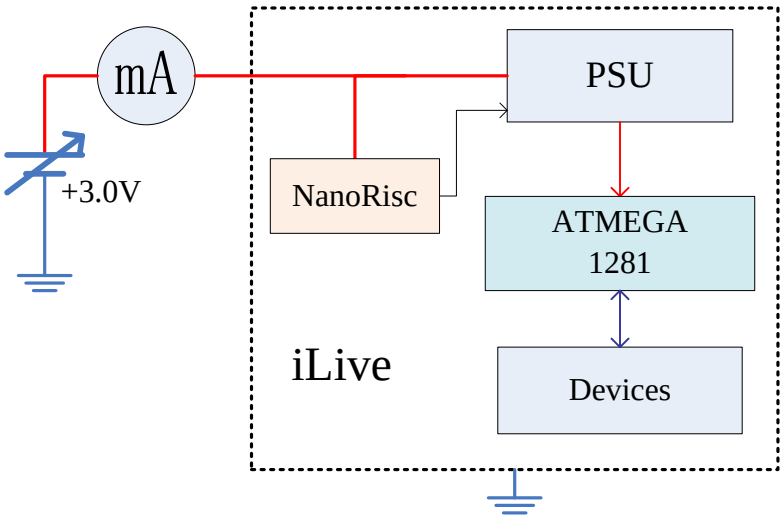

Figure 6-9 Measure Schematics of the iLive

The current of the iLive is also measured when the WSN node handles sensing, signal processing, data storage, and wireless communication task independently. The results are recorded in the Table 6-8 as follow.

Table 6-8 Task Resource Required of iLive

Task	Resource Required		Energy Consumption(μJ)
	Current	Time Used	
Sensing	15.9mA	900ms	42795
Signal Processing	9.9mA	268ms	7919
Data Storage	16.3mA	10.1ms	494
Wireless Communication	21.8mA	140ms	9156
Sleep	1μA	178682ms*	536
Total			60900

*Sleep time is determined by sample frequency. In this dissertation the sample frequency is 3 minutes per sample, which is the same as the TelosB one(J. Polastre et al., 2004).

The total power consumption of iLive can be calculated by (4.3), we can get:

$$\xi(A_{iLive}) = 60900\mu J \ , \tag{6.4}$$

The result is slightly bigger than $\xi^M(A_{AT91SAM7Sx+ATMEGA1281})$ in (6.3). With a pair of AA Lithium/Iron Disulfide (Li/FeS_2) 3000mAh batteries and the sampling frequency is once every 3 minutes, the lifetime of iLive is 1108 days, only 12.76% less than the E-MWSN. However, iLive is much cheaper than the E-MWSN, so this result is acceptable.

6.2.5.2. Reliability

The FIT of NanoRisc is calculated based on datasheet of NanoRisc and FIDES 2009 (ALD, 2012; DGA, September 2010). The FIT of ATMEGA1281 and NanoRisc combine with related external components are listed as follow:

Table 6-9 FIT of each core in the iLive*

Core	FIT
ATMEGA1281	48.07
NanoRisc	13.09

*The raw MTBF or FIT data is taken from manufacturers (Atmel-Corporation, 2012c; Kemet, 2012; Linear, 2009).

From Table 6-9, we can get MTBF of iLive is 1.64E+07 hours or 1866 years. Because iLive does not have redundancy core, so the MTBF and MTBCF of iLive are the same value.

Even the MTBCF of iLive is smaller than the E²MWSN, it still big enough to fulfill the requirement of most applications.

As mentioned in Figure 2-4 and 2.2.1.6, the MTBF/MTBCF is the failure rate (bottom of the bathtub curve) of core components, and it is not the MTBF/MTBCF of the WSN node board. It is also not the product lifetime of WSN node.

6.3. SIS: High Reliability Sensor Node in Smart Irrigation System

6.3.1. General Overview

The Smart Irrigation System (SIS) is a low cost instance of multicore architecture. The objective of the SIS is to implement a low cost, user-friendly and high reliable (safety) multicore WSN node for green house application. It has only two cores of multicore architecture. The Figure 6-10 shows the block diagram of SIS.

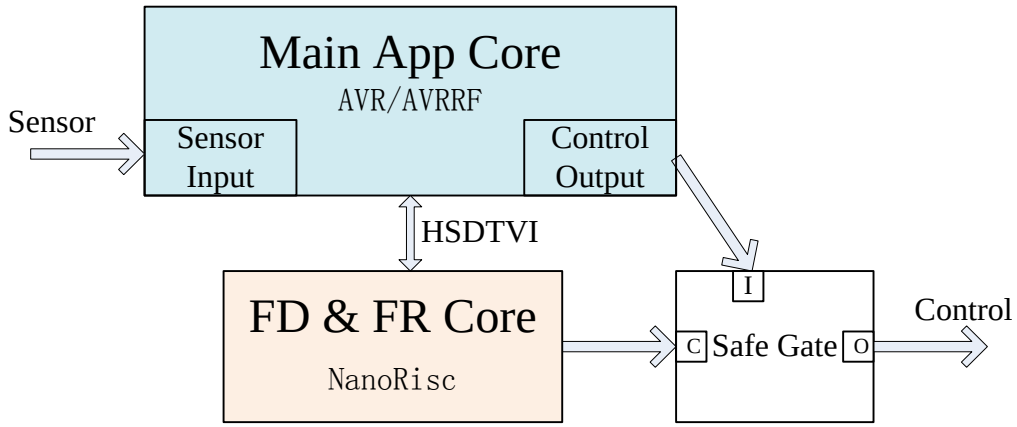

Figure 6-10 Block diagram of the SIS

SIS has two versions: standard version and low price version, the Standard version SIS has IEEE802.15.4 ZigBee wireless access medium, can join WSN to form a bigger and more powerful irrigation system. The low price version removes the IEEE802.15.4 ZigBee wireless access medium in order to archive a lower price. The main different between these two version is the Main App Core. The Standard version uses the same AVRRF as in iLive, and low price version only uses a low cost 8-bit AVR microcontroller.

6.3.2. Hardware Architecture

The Figure 6-11 presents the hardware architecture of the SIS. The SIS supports all previous sensors related to soil such as watermark sensors, decagon sensors and soil temperature sensor. It also supports an external UART port as iLive. Furthermore, it supports an additional buzzer output and a SPDT work mode switch. The buzzer output can generate a sound to inform user when something wrong with system such as battery voltage is too low, water pipe has no water, soil sensor is damage etc. SPDT work mode switch will be used to decide the running mode among Automatic mode, Manual mode, and OFF mode.

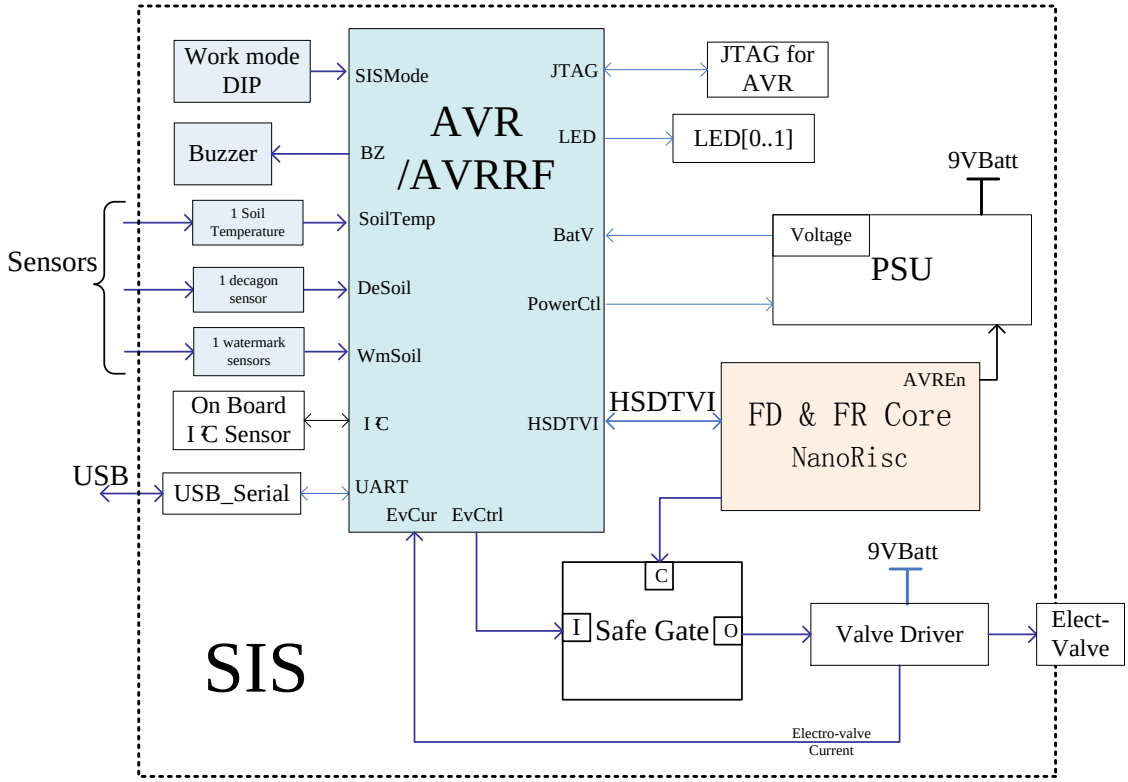

Figure 6-11 Hardware Architecture of the SIS

The most important part in the SIS is the electro-valve driver part. Because the miss control of electro-valve may cause waste of water and sometime flood (safety), so the safety of electro-valve driver is very important. To achieve high safety, the SIS adds a safe gate between normal valve driver and Main App Core AVR/AVRRF. The FD & FR Core NanoRisc will supervise the control process of electro-valve. If any fault is detected during the control process, NanoRisc will force the electro-valve back to close status. This method can greatly improve the safety of the SIS and avoid flood when fault appears. The Figure 6-12 shows the photo of the SIS board.

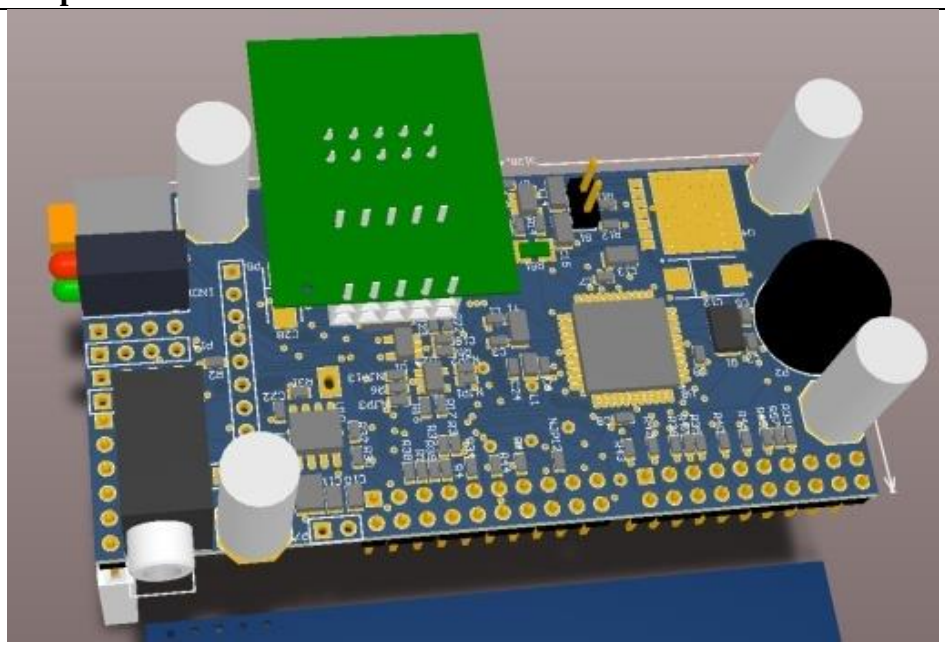

Figure 6-12 Circuit Board of the SIS

6.3.3. Key Features

The main features of the SIS are:

- Low Energy Consumption (1 year Lifetime)
- Reliability and Safety
- 9-Volt Alkaline Battery
- Dimension 79mm*40mm
- 1 DC-9V Electro-valve to control flow of irrigate water
- 1 Buzzer
- 1 Watermark sensors
- 1 Decagon sensors
- 1 soil temperature sensor
- 1 RS232/USB slave port
- Optional
  - 1 air temperature sensor
  - 1 air humidity sensor
  - 1 light sensor
  - IEEE802.15.4 ZigBee wireless access medium.

6.3.4. Performance

6.3.4.1. Power Consumption

To evaluate the power consumption of the SIS, the total running current of the SIS is measured as shown in the Figure 6-13, similar to the Figure 6-4.

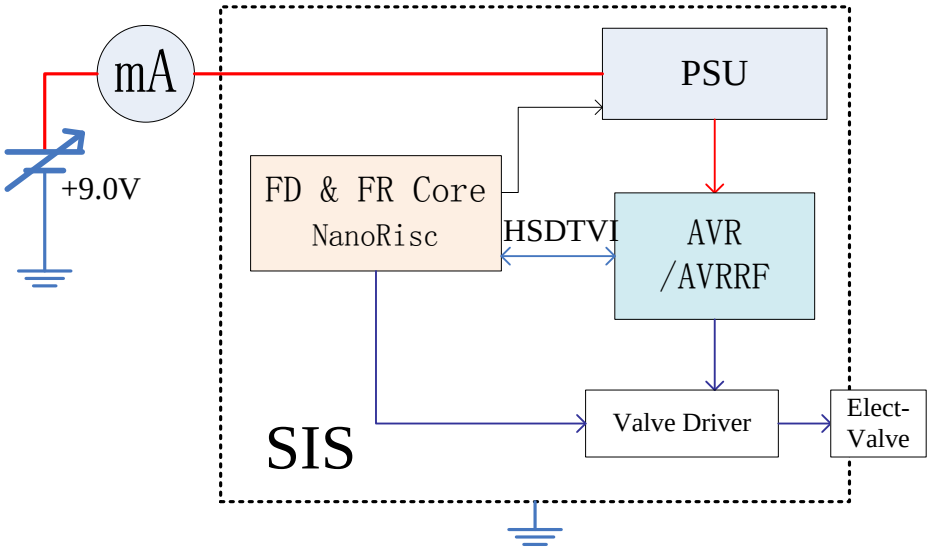

Figure 6-13 Measure Schematics of the SIS

The soil moisture condition may affect the work process of the SIS. When soil moisture is in proper level, after sensing the soil moisture, the SIS will return to sleep without control the electro-valve. While if soil moisture is dry enough (under fixed threshold), SIS will turn on the electro-valve for a while to irrigate the garden. Therefore, the power consumption of the SIS is measured separately based on the normal and dry soil moisture. The results are recorded in the Table 6-10 and the Table 6-11as follow.

Table 6-10 Task Resource Required of the SIS on Dry Soil *

Task	Resource Required		Energy Consumption(μJ)
	Current	Time Used	
Sensing	6.3mA	900ms	51030
Signal Processing	4.6mA	268ms	11095
Data Storage	7.6mA	10.1ms	691
Wireless Communication	15.5mA	140ms	19530
Turn On Electro-valve	650mA	100ms	585000
Turn Off Electro-valve	650mA	100ms	585000

Task	Resource Required		Energy Consumption(μJ)
	Current	Time Used	
Sleep	20μA	178482ms*	32127
Total			1284473

Table 6-11 Task Resource Required of the SIS on Normal Soil

Task	Resource Required		Energy Consumption(μJ)
	Current	Time Used	
Sensing	6.3mA	900ms	51030
Signal Processing	4.6mA	268ms	11095
Data Storage	7.6mA	10.1ms	691
Wireless Communication	15.5mA	140ms	19530
Sleep	20μA	178682ms*	32163
Total			114509

*Sleep time is determined by sample frequency. In this dissertation the sample frequency is 3 minutes per sample, which is the same as the TelosB one(J. Polastre et al., 2004).

The total power consumption of the SIS can be calculated by (4.3), we can get:

$$\xi\left(A_{SIS\_Dry}\right)=1284473\mu J \ , \tag{6.5}$$

$$\xi\left(A_{SIS\_normal}\right)=114509\mu J \ . \tag{6.6}$$

Therefore, if garden need irrigation twice a day, normally the irrigation frequency should be lower than this, the overall power consumption of SIS will be:

$$\xi\left(A_{SIS}\right)=\frac{2\times 3}{1440}\times\xi\left(A_{SIS\_Dry}\right)+\frac{1440-2\times 3}{1440}\times\xi\left(A_{SIS\_Normal}\right)=119384\mu J \ . \tag{6.7}$$

With a Zinc-Manganese Dioxide (Zn/MnO_2) Alkaline 9V 500mAh battery and the sampling frequency is once every 3 minutes, the lifetime of SIS sensor node will be 283 days. However, with the same battery, if the sample frequency is lower, the lifetime of SIS will be longer. If the sample frequency is 10 minutes per sample, the lifetime of SIS will be increase to 577 days, which can exceed the 1-year lifetime requirement.

6.3.4.2. Reliability

The FIT of AVRRF and NanoRisc combine with related external components are listed as follow:

Table 6-12 FIT of Each core in the SIS*

Core	FIT
AVRRF	51.70
NanoRisc	13.09

*The raw MTBF or FIT data is taken from manufacturers (Atmel-Corporation, 2012c; Kemet, 2012; Linear, 2009).

From Table 6-12, we can get MTBF of SIS is 1.54E+07 hours or 1762 years, big enough to fulfill the requirement of greenhouse applications.

Besides, AVRRF and NanoRisc is parallel backup of each other in electro-valve control. Therefore, the MTBCF of core components in SIS for electro-valve control is 1.48E+15 hours, 1.69E+11 years, which make the SIS especially safe.

As mentioned in Figure 2-4 and 2.2.1.6, the MTBF/MTBCF is the failure rate (bottom of the bathtub curve) of core components, and it is not the MTBF/MTBCF of the WSN node board. It is also not the product lifetime of WSN node.

6.4. iLiveEdge: High Reliability and Multi-Support Multicore WSN Edge Router

6.4.1. General Overview

The iLiveEdge is a variation of original multicore WSN architecture. The objective of the iLiveEdge is to implement a low cost and high reliable multi-support multicore WSN edge router for all WSN applications. The iLiveEdge has three cores, the Main App Core in the iLiveEdge is an AVRRF for local WSN access, the Auxiliary Core in the iLiveEdge is an AT91SAM7Sx for Internet access, and the FD & FR Core is a NanoRisc. In the iLiveEdge, AVRRF and AT91SAM7Sx are both always active; they collaborate to work as the bridge of local WSN and global Internet server. The Figure 6-14 shows the block diagram of the iLiveEdge.

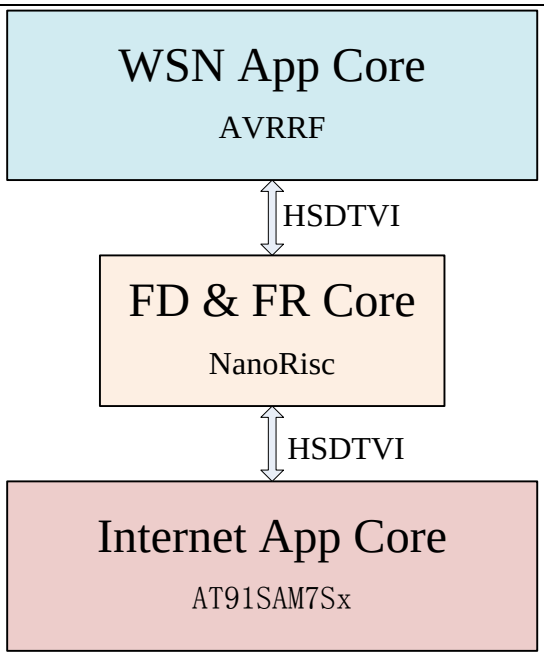

Figure 6-14 Block diagram of the iLiveEdge

6.4.2. Hardware Architecture

The Figure 6-15 presents the hardware architecture of the iLiveEdge. The WSN App Core AVRRF will be the coordinator of local WSN. It exchanges data with other WSN nodes through IEEE802.15.4. The Internet App Core AT91SAM7Sx supports different methods such as Ethernet, WiFi, GPRS and 3G for Internet access. The AVRRF and AT91SAM7Sx will exchange data through an UART. The FD & FR Core NanoRisc will run as a monitor for both AVRRF and AT91SAM7Sx in order to improve the reliability of the iLiveEdge.

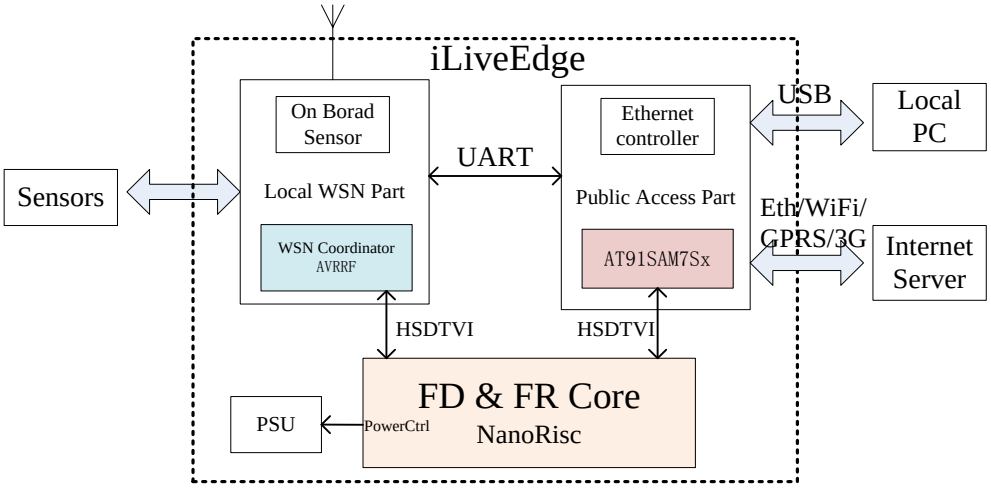

Figure 6-15 Hardware Architecture of the iLiveEdge

The Figure 6-16 shows the photo of the iLiveEdge board.

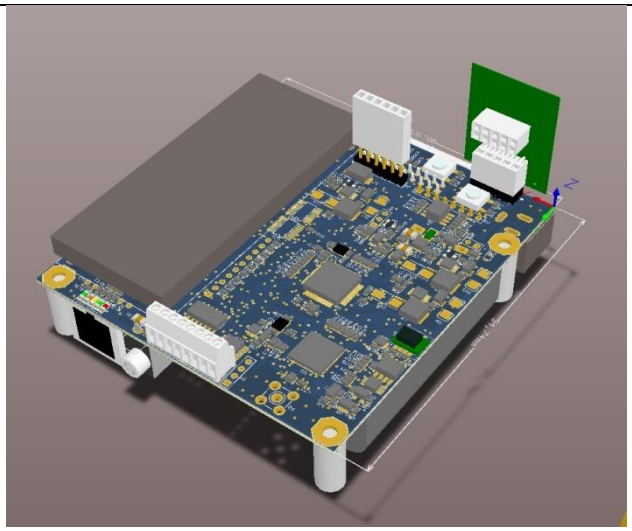

Figure 6-16 CircuitBoard of the iLiveEdge

6.4.3. Key Features

The major features of the iLiveEdge are:

- Dimension 95mm*88mm
- 1 light sensor
- 1 temperature sensor
- 1 air humidity sensor
- 4 Watermark soil moisture sensors
- 1 Decagon soil moisture sensor
- 1 RS232 ports
- 1 USB slave port
- Multi-Support for Ethernet, WiFi, Bluetooth and GPRS
- IEEE802.15.4 ZigBee wireless access medium

6.4.4. Performance

6.4.4.1. Power Consumption

The power consumption is not the key parameter of the iLiveEdge. For the Local WSN part in the iLiveEdge is coordinator, normally this part cannot sleep. In order to provide continuous services, the iLiveEdge requires continuous power supply such as AC-DC on electricity grid or big rechargeable battery with renewable power generators.

6.4.4.2. Reliability

The FIT of each core in the iLiveEdge combine with related external components are listed as follow:

Table 6-13 FIT of Each core in the iLiveEdge*

Core	FIT
AVRRF	51.70
NanoRisc	13.09
AT91SAM7Sx	58.93

*The raw MTBF or FIT data is taken from manufacturers (Atmel-Corporation, 2012c; Kemet, 2012; Linear, 2009).

Because the iLiveEdge does not have redundancy core, so the MTBF and MTBCF of the iLiveEdge are the same. From the Table 6-13, we can get the MTBF of the iLiveEdge is 8.08E+06 hours or 923 years, big enough to fulfill the requirement of most applications.

As mentioned in Figure 2-4 and 2.2.1.6, the MTBF/MTBCF is the failure rate (bottom of the bathtub curve) of core components, and it is not the MTBF/MTBCF of the WSN node board. It is also not the product lifetime of WSN node.

6.5. EPER: Highest Performance High Reliability and Multi-Support Multicore WSN Edge Router

6.5.1. General Overview

The Extend PandaBoad Edge Router EPER is also a variation of original multicore architecture. It is an upgrade edge router for high performance application. The Internet core of the iLiveEdge is a 32-bit RISC ARM7 AT91SAM7Sx, which runs at 48 MHz and only has 512-Kbyte of flash program memory and 64-Kbyte static RAM. This core is not powerful enough; the storage is also not big enough. For some applications such as smart stick, inter vehicle communication, the edge router is expected to handle multimedia data stream in real-time. Therefore, we develop a new powerful edge router based on the PandaBoard, which has a Dual-core 1.2 GHz ARM A9 chip with 1GB RAM. The Figure 6-17shows the block diagram of the EPER.

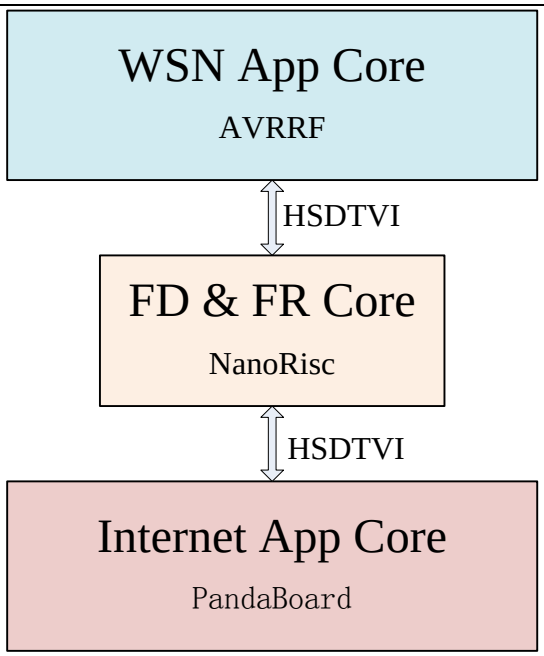

Figure 6-17 Block diagram of the EPER

6.5.2. Hardware Architecture

The Figure 6-18 presents the hardware architecture of the EPER. The WSN App Core AVRRF will be the coordinator of local WSN. It exchanges data with other WSN nodes through IEEE802.15.4. The Internet App Core is based on the PandaBoard, which supports different methods such as Ethernet, WiFi, GPRS and 3G for Internet access. The AVRRF and the PandaBoard will exchange data through an UART. The FD & FR Core NanoRisc will run as a monitor for both the AVRRF and the PandaBoard in order to improve the reliability of the EPER.

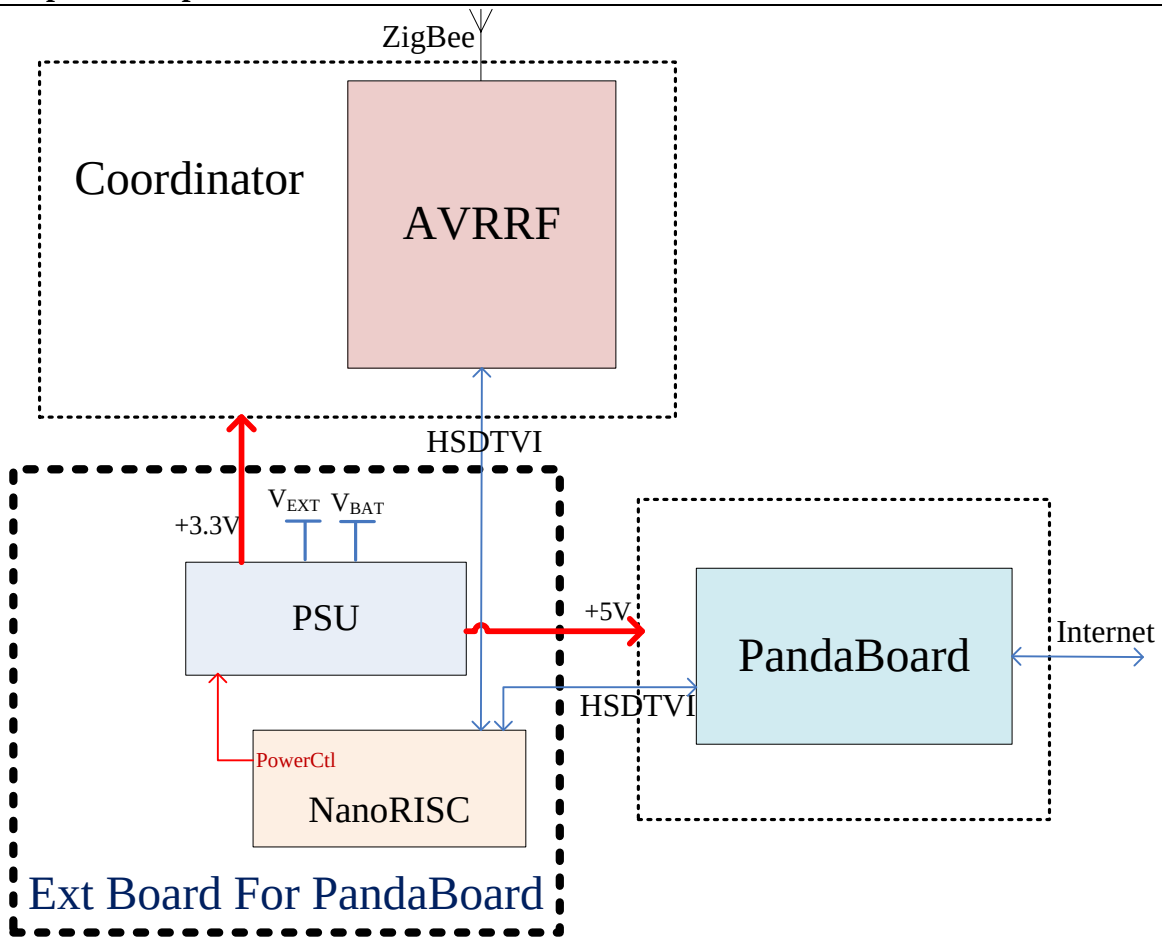

Figure 6-18 Hardware Architecture of the EPER

The Figure 6-19 shows the photo of the EPER board.

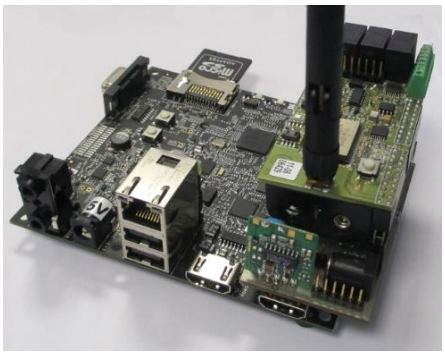

Figure 6-19 Circuit Board of the EPER

6.5.3. Key Features

The main features of the EPER are:

- High Reliability

- Highest Performance
- Dual-core 1.2 GHz ARM® Cortex™-A9 MPCore™ with SMP
- 1 GB low power DDR2 RAM
- Dimension 114mm*101mm
- 1 light sensor
- 1 temperature sensor
- 1 air humidity sensor
- 1x USB 2.0 HS OTG port
- 2x USB 2.0 HS Host ports
- 1 RS232 ports
- Multi-Support for Ethernet, WiFi, Bluetooth
- IEEE802.15.4 ZigBee wireless access medium

6.5.4. Performance

6.5.4.1. Power Consumption

For the same reason, the power consumption is not the key parameter of the EPER. In order to provide continuous services, the EPER also need have continuous power supply such as AC-DC on electricity grid, big rechargeable battery with renewable power generators, etc.

6.5.4.2. Reliability

The FIT of each core in EPER combine with related external components are listed as following:

Table 6-14 FIT of Each core in EPER*

Core	FIT
AVRRF	51.70
NanoRisc	13.09
PandaBoard	420.24

*The raw MTBF or FIT data is taken from manufacturers (Atmel-Corporation, 2012c; Kemet, 2012; Linear, 2009).

Because the EPER does not have redundancy core, so the MTBF and MTBCF of EPER are the same. From Table 6-14, we can get MTBF of EPER is 2.06E+06 hours or 235 years, big enough to fulfill the requirement of most applications.

As mentioned in Figure 2-4 and 2.2.1.6, the MTBF/MTBCF is the failure rate (bottom of the bathtub curve) of core components, and it is not the MTBF/MTBCF of the WSN node board. It is also not the product lifetime of WSN node.

6.6. RPiER: Higher Performance High Reliability and Multi-Support Multicore WSN Edge Router

6.6.1. General Overview

The Raspberry Pi Based Edge Router is also a variation of original multicore architecture. It is a low cost edge router version comparing with the EPER one. The Internet core of the RPiER is based on the Raspbery Pi Board, which has a 32-bit 700 MHz ARM1176JZF-S core (ARM11 family) and 512MB of DRAM. This core is powerful enough to handle most of our applications. The Figure 6-20 shows the block diagram of the RPiER.

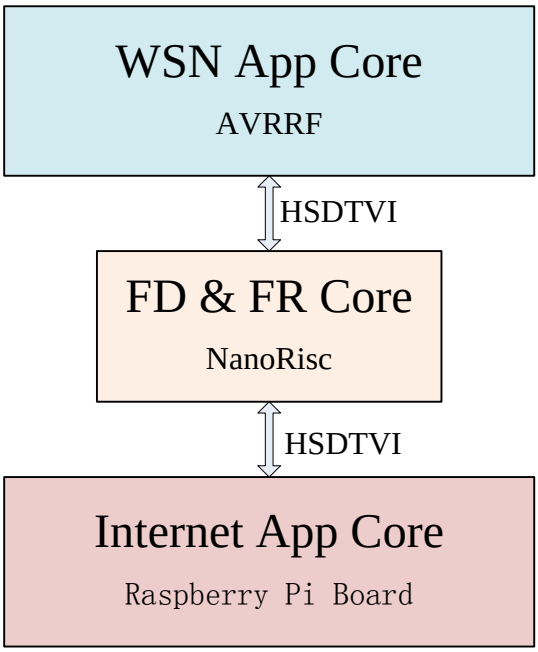

Figure 6-20 Block diagram of the RPiER

6.6.2. Hardware Architecture

The Figure 6-21 presents the hardware architecture of the RPiER. The WSN App Core AVRRF will be the coordinator of local WSN. It exchanges data with other WSN nodes

through IEEE802.15.4 wireless access medium. The Internet App Core Raspberry Pi board supports different methods such as Ethernet, WiFi, GPRS and 3G for Internet access. The AVRRF and the Raspberry Pi board will exchange data through an UART. The FD & FR Core NanoRisc will run as a monitor for both the AVRRF and the Raspberry Pi board in order to improve the reliability of the RPiER.

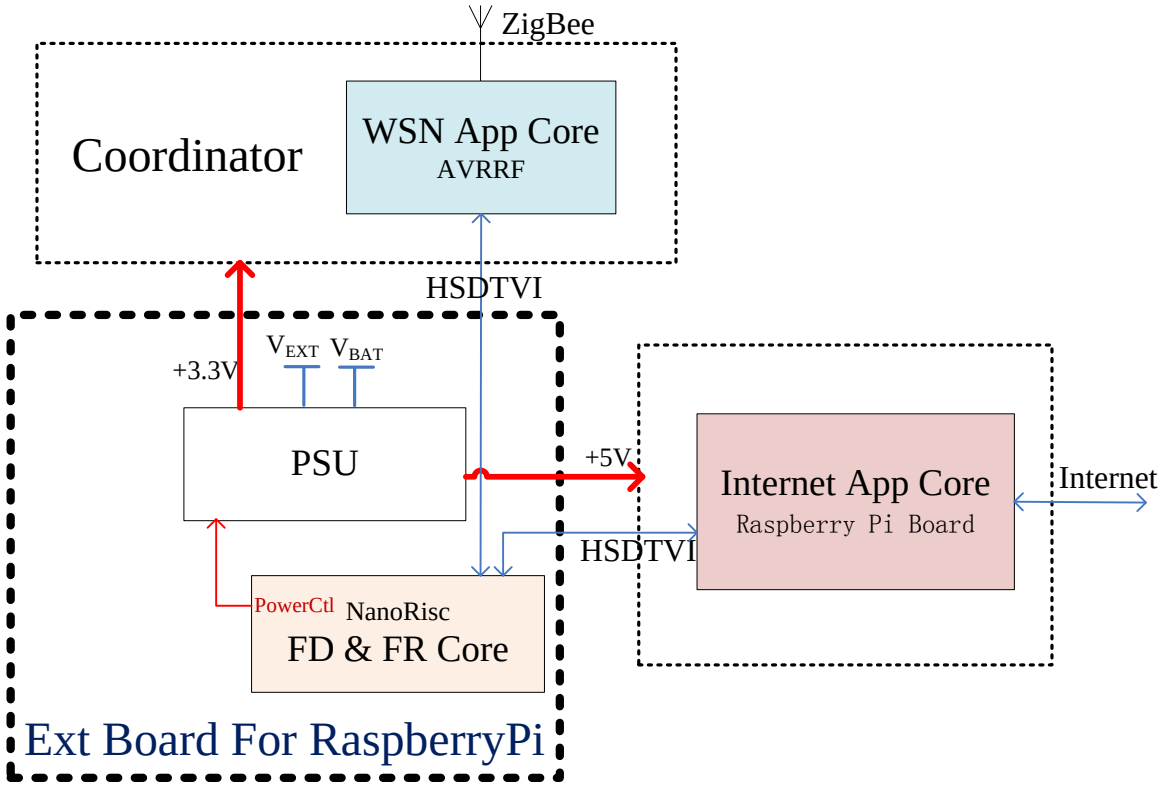

Figure 6-21 Hardware Architecture of the RPiER

The Figure 6-22 shows the photo of the RPiER board.

Figure 6-22 CircuitBoard of RPiER

6.6.3. Key Features

The major features of the RPiER are:

- High Reliability
- Higher Performance
- 700 MHz ARM1176JZF-S core (ARM11 family)
- 256/512MB low power DDR2 RAM
- Dimension 86mm*54mm
- 1 light sensor
- 1 air temperature sensor
- 1 air humidity sensor
- 2x USB 2.0 HS Host ports
- SD / MMC / SDIO card slot
- Composite RCA (PAL & NTSC), HDMI, LCD Panels via DSI
- Multi-Support for Ethernet, WiFi, Bluetooth
- IEEE802.15.4 ZigBee wireless access medium

6.6.4. Performance

6.6.4.1. Power Consumption

For the same reason, the power consumption is not the key parameter of the RPiER. In order to provide continuous services, the RPiER also requires continuous power supply such as AC-DC on electricity grid or big rechargeable battery with renewable power generators.

6.6.4.2. Reliability

The FIT of each core in the RPiER combine with related external components are listed as following:

Table 6-15 FIT of Each core in RPiER*

Core	FIT
AVRRF	51.70
NanoRisc	13.09
Raspberry Pi	249.94

*The raw MTBF or FIT data is taken from manufacturers (Atmel-Corporation, 2012c; Kemet, 2012; Linear, 2009).

Because the RPiER does not have redundancy core, so the MTBF and MTBCF of the RPiER are the same. From Table 6-15, we can get the MTBF of the RPiER is 3.18E+06 hours or 363 years, big enough to fulfill the requirement of most applications.

As mentioned in Figure 2-4 and 2.2.1.6, the MTBF/MTBCF is the failure rate (bottom of the bathtub curve) of core components, and it is not the MTBF/MTBCF of the WSN node board. It is also not the product lifetime of WSN node.

6.7. Related Projects

Based on the previous WSN nodes, SMIR team has carried out several real world IoT and WoT projects. The Table 6-16 provides a list of these projects.

Table 6-16 Related Ongoing IWoT Real World Projects in SMIR@LIMOS

Index	Project Field	Project Name	Project Type	Comment
1	*Precision Agriculture*	iLive, MiLive	Scientific Cooperation Project	
2	*Green House*	Smart Irrigation System	Innovative Project	
3	*Smart Care*	Smart Environment Explorer Stick	Innovative Project	

We will discuss each project in following parts.

6.7.1. Precision Agriculture

There mainly two types of Precision Agriculture platform in our group: scalar WSN platform and multimedia WSN platform.

6.7.1.1. Scalar WSN platform: iLive

As mentioned before, the iLive board is a scalar WSN node dedicated to environment data collection and precision agriculture. The iLive directly supports many environmental sensors: 4 Watermark soil moisture sensors, 3 Decagon soil moisture sensors, 1 air temperature sensor, 1 soil temperature sensor, 1 air humidity sensor and 1 light sensor. It has an ultra low power nano-controller and an 8-bit RISC AVR microprocessor. The iLive node is a standard WSN node having embedded IEEE802.15.4 transceiver on board. A set of the iLive nodes can work together and build a scalar WSN. The iLive has a RS232/USB slave port which may be used to connect to a PC or a Raspberry Pi Board. The iLive has an extension connector having I²C, SPI, ADC and GPIO interfaces which can be used to add specific sensors or devices when necessary.

You also can visit our long-term the iLive online demo on http://edss.isima.fr/. The demo has been continuously operating for more than one year. The login username and password are both "demo".

Following are some photos related to the iLive platform.

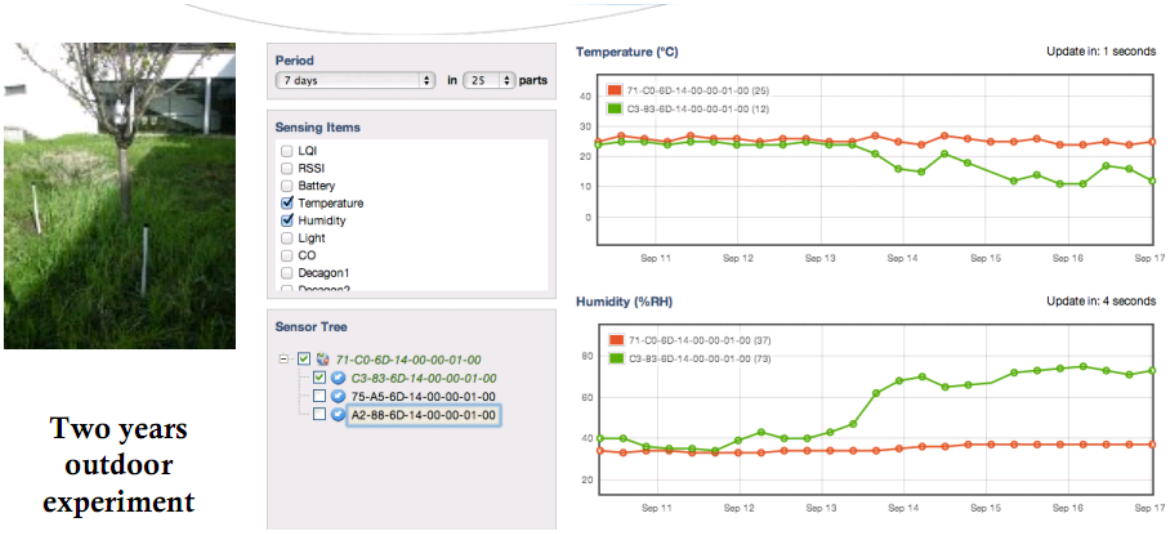

Figure 6-23 Outdoor Experiment in ISIMA Garden

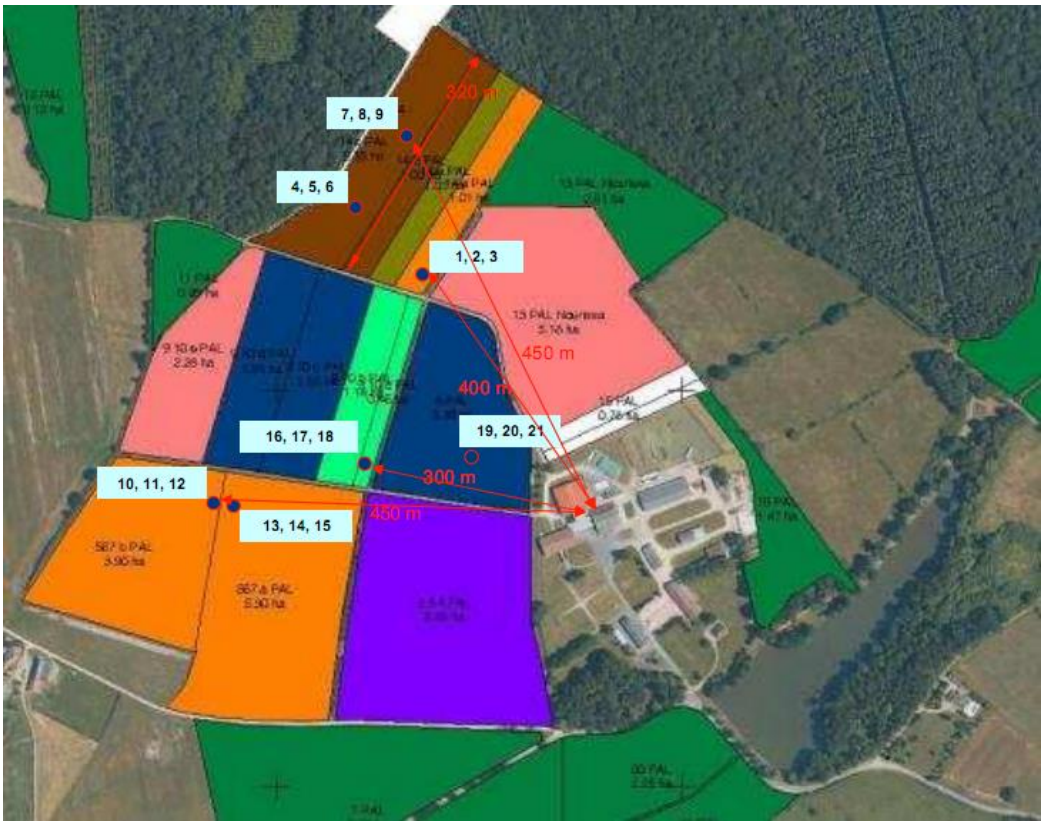

Figure 6-24 Real world Experiment in Montoldre (Cooperation with Irstea)

6.7.1.2. Multimedia WSN platform: MiLive

The MiLive is a multicore multimedia WSN node. It is built around 2 boards (size= 76mm*40mm): scalar WSN node (iLive) and Wireless Multimedia node based on credit card format Raspberry Pi (MWiFi).

Figure 6-25 shows the heterogeneous architecture of the MiLive.

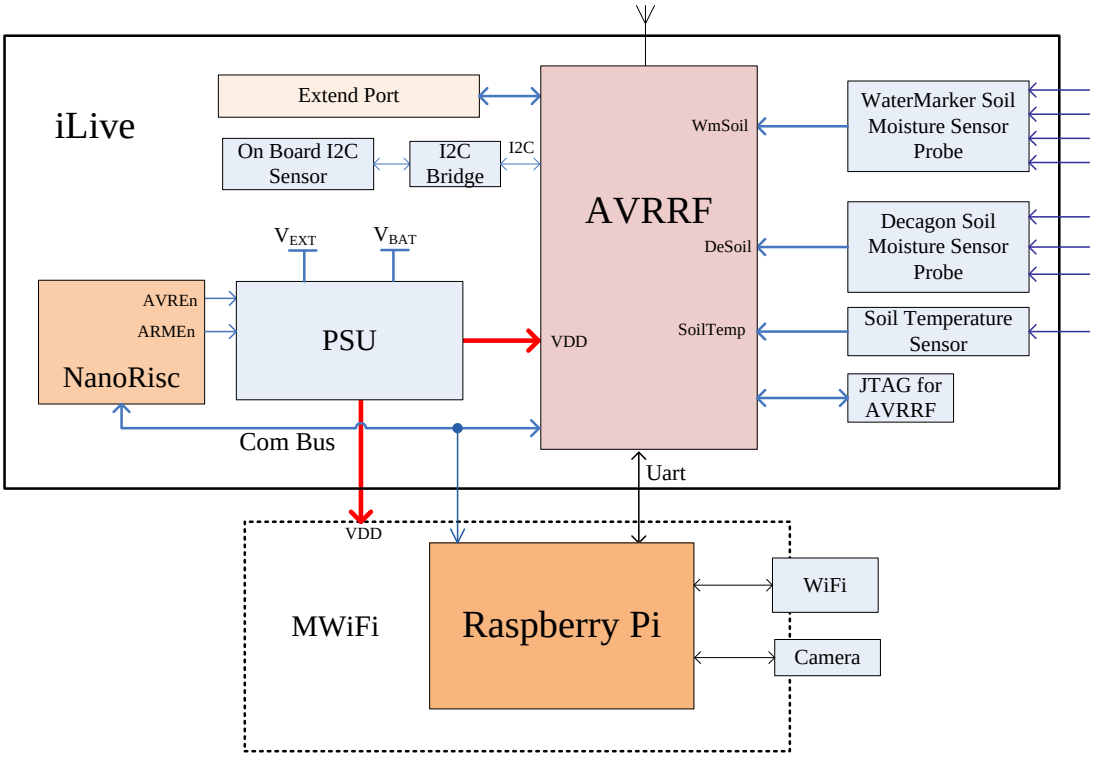

Figure 6-25 Heterogeneous Architecture of the MiLive

You can visit the demo for the MiLive platform on http://edss.isima.fr/demoforall/. The login username and password are also both "demo".

Following are some photos related to the MiLive platform.

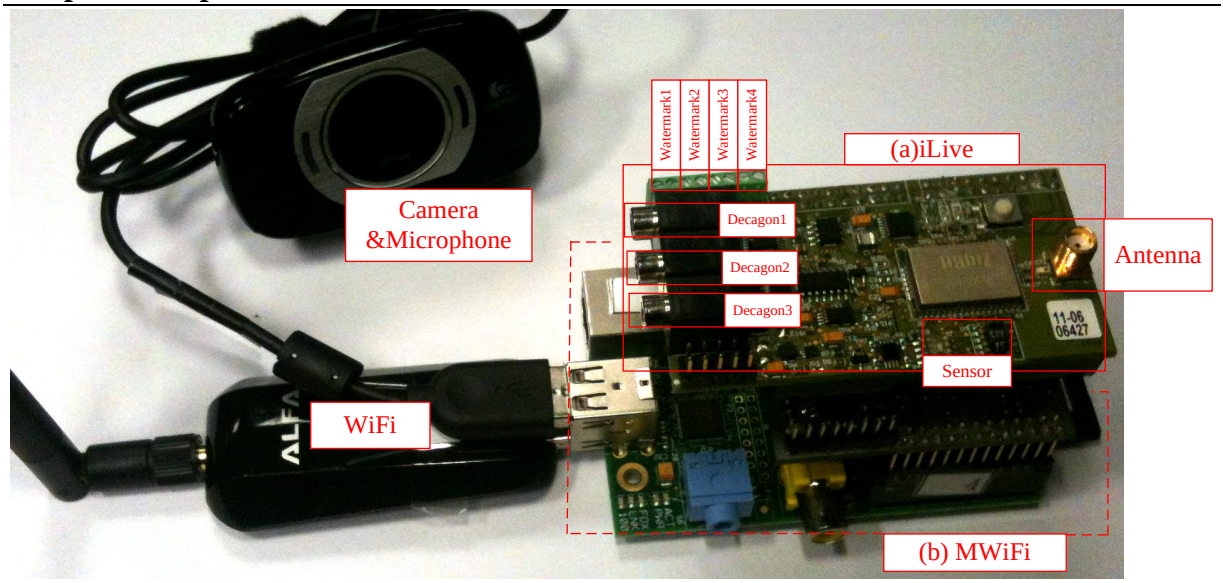

Figure 6-26 Circuit Board of the MiLive

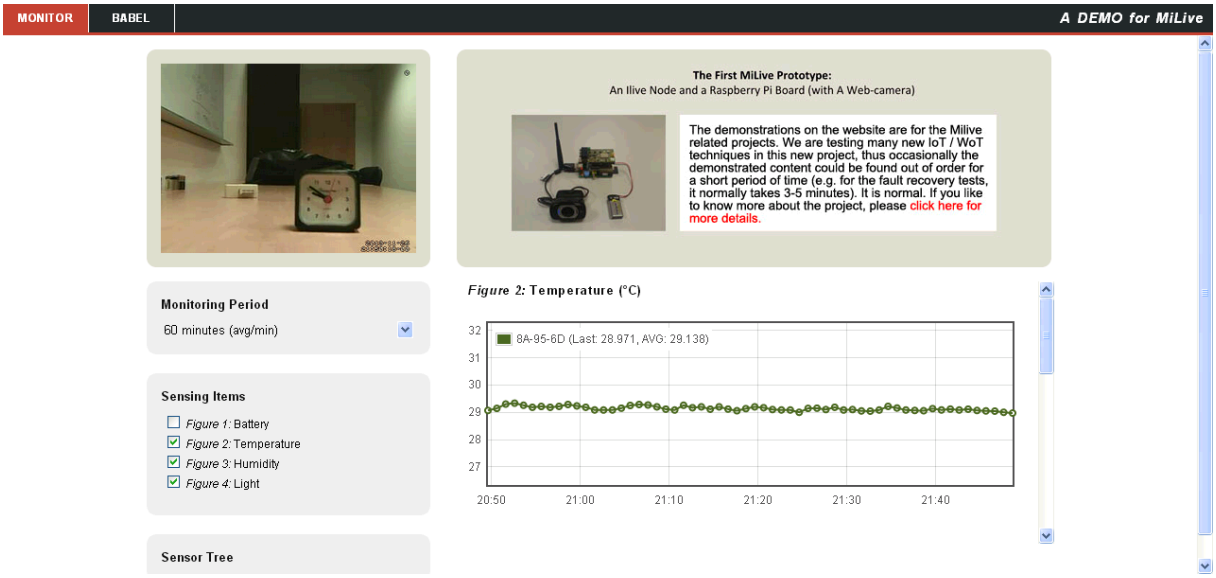

Figure 6-27 Demo Web page of the MiLive platform

6.7.2. Smart Irrigation System

The SIS (Smart Irrigation System) is a new irrigation technology based on remotely configurable wireless embedded system. It is a total solution including from the low-cost reliable sensor board to the user-friendly user interface.

It provides with a cooperative and automotive irrigating mechanism that helps less-knowledge planters growing plants and save water resource. It is provided with the reliable multi-support hardware components.

With the hardware supports, the SIS can be customized to adapt to different network scenarios such as small gardens, greenhouses, football fields and large farms.

Currently, the integration interface of the SIS is still in the development stage, but a simple demo is available on http://edss.isima.fr/sites/smir/sis. You can try to control remotely the watering devices through Internet.

Following are some photos related to the SIS platform.

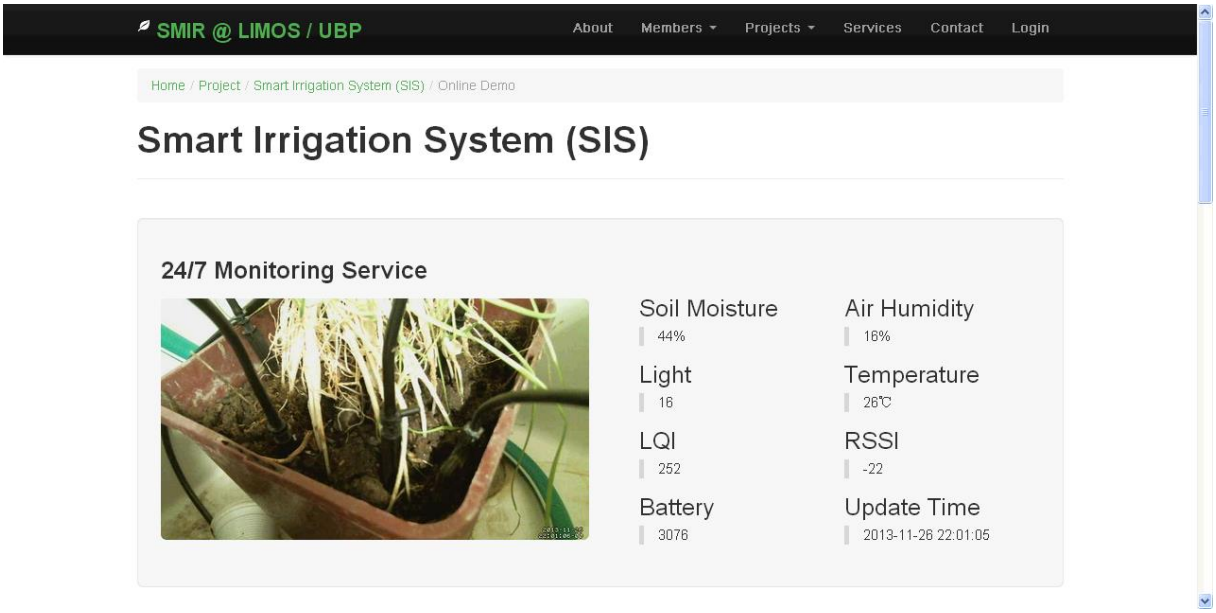

Figure 6-28 Demo Web page of the SIS Platform

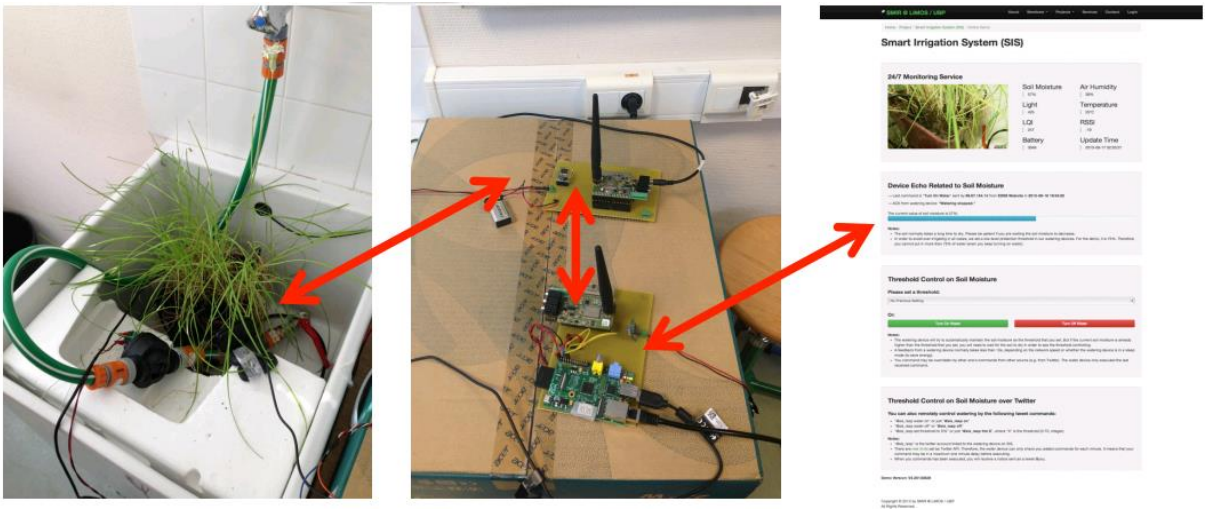

Figure 6-29 Real World Long Term Online Demo of the SIS Platform

6.7.3. Smart Environment Explorer Stick

The Smart Environment Explorer Stick (SEES) is project to develop an enhanced smart white cane, which assists the Visually Impaired Person or People (VIP)'s navigation. The active multi-sensor context-aware concept is adopted to be implemented in the SEES to help the VIP to move safely and easily in any places in the world (indoor or outdoor).

The Figure 6-30 shows the architecture of SEE-stick. The SEE-stick will use multicore WSN architecture, which has two cores.

One core is the Raspberry Pi board, which will work as CPU (central processing unit) to handle the complex tasks. The other core is an 8-bit RISC microprocessor AVRRF, which will handle some scalar sensors. The two cores connect each other through a Hardware Support Debug Test and Validate Interface (HSDTVI).

Through HSDTVI, the Raspberry Pi board and AVRRF can mutually check their running status in real-time. When any critical fault occurred in one of two cores, the other core can detect it, and handle it with appropriate actions, which can help SEE-stick to recover from fault or generate alarm to inform VIP to stay in safe state.

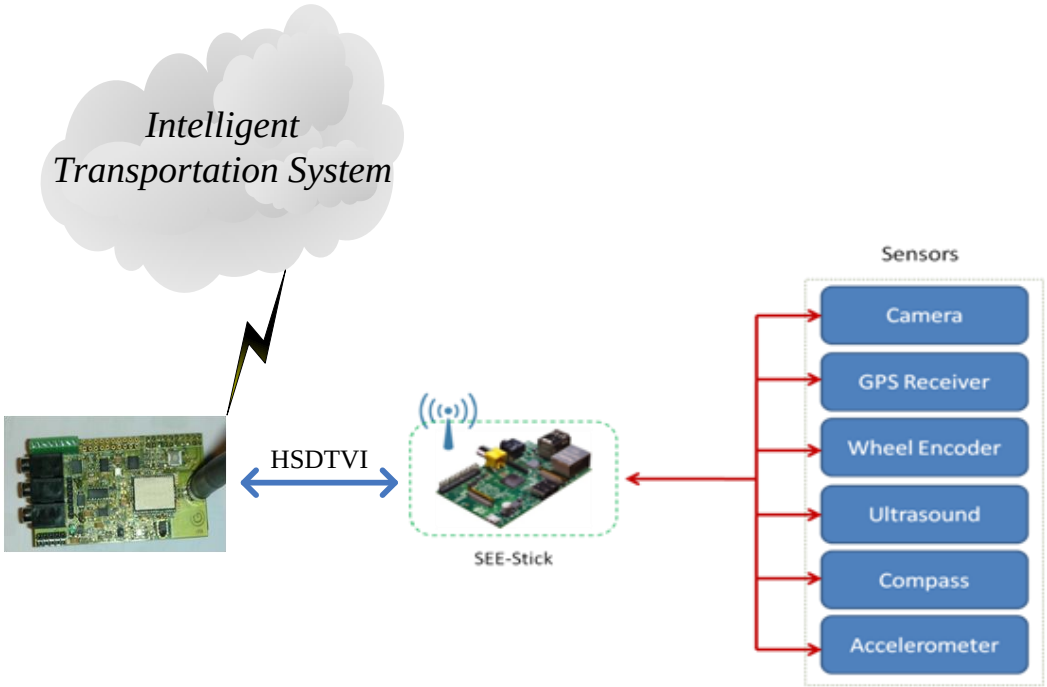

Figure 6-30 Block diagram of the SEE-stick

Besides, the SEE-stick supports many wireless access media, such as WiFi, 3G, and IEEE 802.15.4 etc. The IEEE 802.15.4 RF part, supported by AVRRF core, will provide the SEE-stick to access local ITS 'Intelligent Transportation System', which can greatly help the SEE-stick to provide more accurate and reliable mobility cues.

The SEE-stick will run in a real word, unpredictable, physical environment, which makes faults inevitable. In order to provide more reliable outputs (mobility cues), even in the presence of faults, we have to improve dependable of our SEE-stick in every part. We expect that the checking and recovering mechanism on multicore and multi-support wireless can greatly help to build a robust SEE-stick.

Currently, the SEE-stick is still in development stage, but there is a remote monitoring demo available on http://edss.isima.fr/sites/smir/sees. You can see the last navigation of SEE-stick by this demo.

Following are some photos related to the SEE-Stick.

Figure 6-31 SEE-Stick Prototype

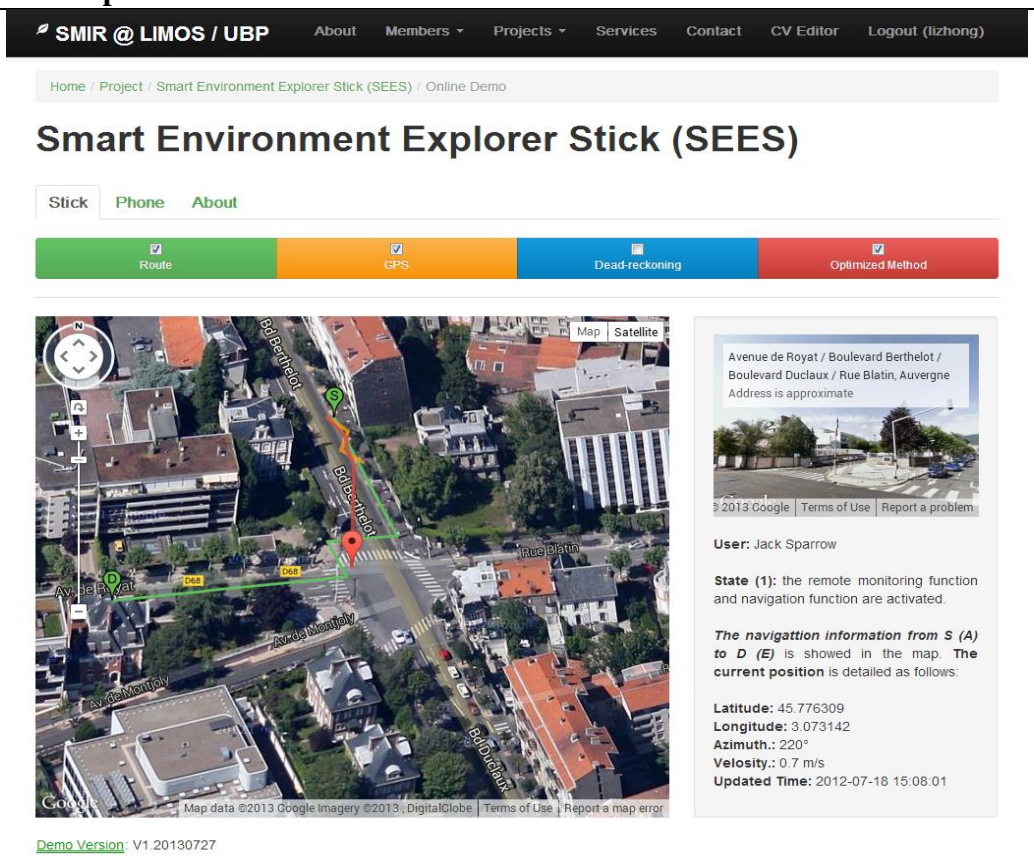

Figure 6-32 SEE-Stick Remote Monitoring Demo Web page

6.8. Summary

We develop, test and validate several WSN nodes based on multicore architecture. The Table 6-17 provides the key features of all multicore WSN nodes mentioned before.

Table 6-17 Key Features of Different Multicore WSN Nodes

Index	Feature Name	Multicore WSN Nodes					
		E^2MWSN	iLive	SIS	iLiveEdge	EPER	RPiER
1	IEEE802.15.4	★	★	★	★	★	★
2	Air Temperature	★	★	★	★	★	★
3	Air Humidity	★	★	★	★	★	★
4	Light Sensor	★	★	★	★	★	★
5	UART	★	★	★	★	★	★
6	Decagon Soil Moisture	★	★	★			
7	Watermark Soil Moisture		★	★			
8	Soil Temperature		★	★			

Index	Feature Name	Multicore WSN Nodes					
		E^2MWSN	iLive	SIS	iLiveEdge	EPER	RPiER
9	Elector-Valve			★			
10	Passive Infrared Type Motion Detector	★					
11	SD Card	★				★	★
12	USB Host Port					★	★
13	USB Client Port	★				★	
14	Ethernet				★	★	★
15	WiFi				★	★	★
16	Bluetooth				★	★	★
17	GPRS Modem				★	★	★
18	3G Modem				★	★	★
19	Camera					★	★
20	Microphone					★	★

The Table 6-18 provides the reliability of all multicore WSN nodes mentioned before. From the Table 6-18 we can find that the reliability of multicore WSN node is very good, the MTBF/MTBCF of nodes can up to 5000 years. Even for complex edge router, the MTBF/MTBCF of ER is also above 200 years, big enough for most of the applications.

As mentioned in Figure 2-4 and 2.2.1.6, the MTBF/MTBCF is the failure rate (bottom of the bathtub curve) of core components, and it is not the MTBF/MTBCF of the WSN node board. It is also not the product lifetime of WSN node.

Table 6-18 Reliability of Different Multicore WSN Nodes

Index	Multicore WSN Nodes	FIT of Cores			MTBF (Hours)	MTBF (Years)	MTBCF (Hours)	MTBCF (Years)
		AppCore	AuxCore	FD&FRCore				
1	E^2MWSN	48.07	58.93	22.39	7.73E+06	882	4.47E+07	5099
2	iLive	48.07		13.09	1.64E+07	1866	1.64E+07	1866
3	SIS	51.70		13.09	1.54E+07	1762	1.54E+07	1762
4	iLiveEdge	51.70	58.93	13.09	8.08E+06	923	8.08E+06	923
5	EPER	51.70	420.24	13.09	2.06E+06	235	2.06E+06	235
6	RPiER	51.70	249.94	13.09	3.18E+06	363	3.18E+06	363

Chapter 7. Conclusion and Ongoing Work

7.1. Conclusion

In this dissertation, we have presented a multicore architecture for the design of fault tolerance wireless sensor networks. By introducing NanoRisc, HSDTVI, and standby sparing fault tolerant mechanism, multicore architecture can highly improve the reliability of WSN without significantly increasing cost and complexity. Multicore architecture is capable of addressing the dependability and lifetime requirements of wireless sensor networks. The developed hardware platforms and the real-world applications have already validated the effectively of our multicore WSN architecture.

We also developed a design process (High Reliability Design Process dedicated to High Resource Constraint Embedded System: HRDP) based on multicore architecture to ease the development of high reliable embedded product. By applying HRDP in our real-world projects, we show that HRDP can help us in every design stage: architecture design, early validation, debugging and testing.

To validate the flexibility of our multicore architecture, we developed several hardware platforms, such as E²MWSN, iLive, SIS, iLiveEdge, EPER and RPiER. These hardware platforms are instances of multicore architecture with varying degrees of hardware complexity. We show that the multicore architecture not only improves WSN node lifetime and robustness but also enables to meet diverse application domains by exploring context-aware approach (resource-aware) and local and distributed collaborative processing. These platforms have proven themselves both in theory and through deployment in long-term, battery operated real-world applications.

WSN is an emergent and multidisciplinary science, which is a very active and competitive research field, and is considered as a key technology of the 21st century . In spite of its unlimited potential applications, currently it still works in non-mission critical application. The main obstacle of applying WSN in mission critical applications, such as smart care and real-time industrial process control is that those applications demand extremely high levels of reliability and safety. The multicore architecture presented in the dissertation is ready to meet the demands of real-world mission critical applications. We hope that multicore architecture will contribute significantly to the progress of IoT and WoT evolution.

7.2. Next Generation WSN node

7.2.1. Next Generation Multicore SoC for WSN node

In order to meet different requirements of different applications, we implemented many hardware boards, such as E²MWSN, iLive, SIS, iLiveEdge, EPER and RPiER. These COST solutions increase the complexity of both implementation and maintenance. The cost and size of COST solution are also not optimal and flexible, which enable to implement easily black box concept. Therefore, for the next generation WSN node, we suggest to implement WSN node based on multicore SoC chip. This section will discuss the black box concept and some technical schemes related to Multicore SoC chip, which is particularly optimized for WSN platform.

7.2.1.1. Black box concept

As we mentioned before, in current COST solution, we need to implement many boards for different projects, even these implementations adopt similar multicore architectures. We want to simplify the new design of WSN based on SoC chip. Therefore, we propose to implement black box concept in the SoC chip.

The black box concept means that the SoC chip will be highly configurable. It will be very flexible that it can be used in different application with only one chip. The user just needs download different configuration firmwares, the chip will change the running mode to meet the application requirements.

The IoT applications are unlimited, so it is very important to ease the development of a new project. Following black box concept, we can easily meet multi-project requirements to decrease the Time to Market (TTM), lower the total cost, reuse the technical resource, minimize the development cost for the new project, and ease the stock management.

Besides, if one application core is suffering from permanent fault, it is possible to reconfigure the SoC to fix some permanent faults remotely. This can also greatly ease the maintenance.

The Figure 7-1 shows the architecture of next generation multicore SoC for WSN nodes. It's similar as the COST multicore WSN architecture presented in the Figure 4-2. The different between the Figure 7-1 and the Figure 4-2 is that in the SoC the key components will have redundancy to enable the fault tolerant. These redundant cores will be active by the configuration. If they are not active, the redundant cores will be totally power off, to lower the total energy consumption.

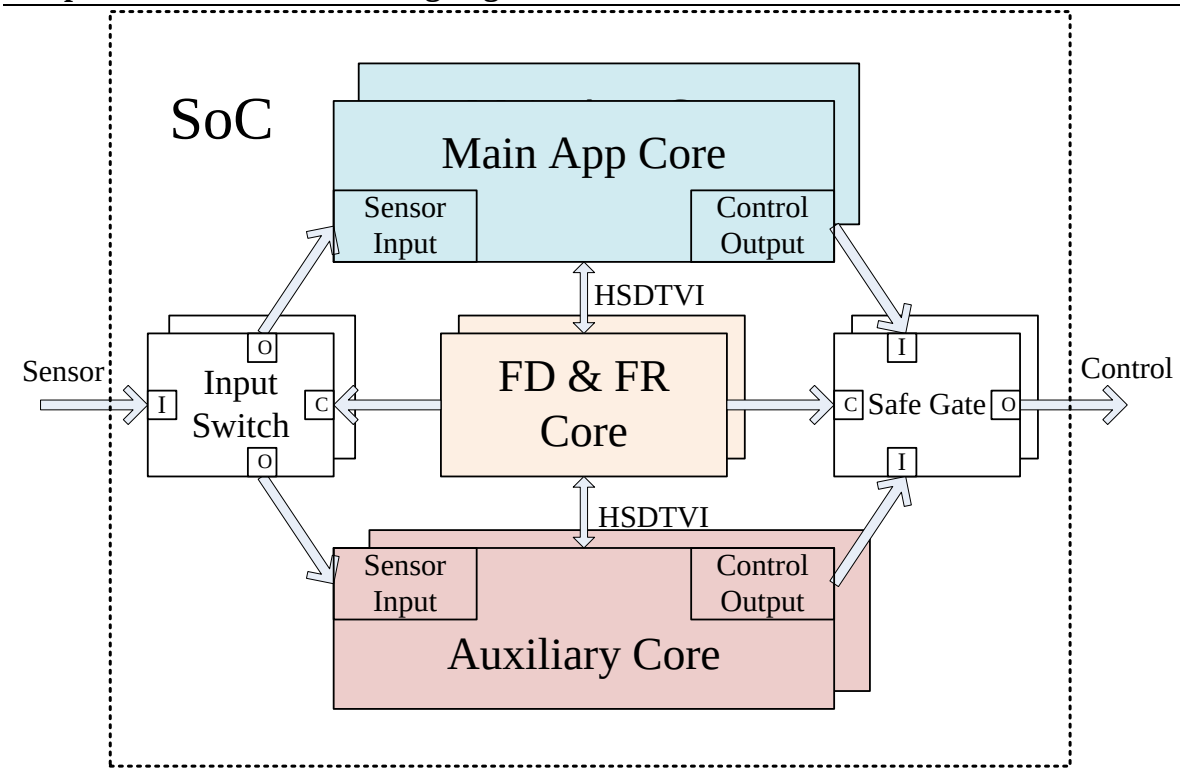

Figure 7-1 Architecture of Next Generation Multicore SoC

7.2.1.2. Chosen of Core

The cores in the Figure 7-1 have several choices: Uniform NanoRisc Array, Different cell (4-bit, 8-bit, 16-bit, 32-bit), or FSMOS Modules.

7.2.1.2.1. Uniform NanoRisc Array

In this method, the core processor of SoC is consisting by an array of NanoRisc. This method eases the implementation of SoC, but increases the complexity of the configuration. The software will also need to divide into pieces in order to run as distributed parts. The minimum power control cell is a NanoRisc, so the power efficiency will be very high. The architecture of this method is shown in the Figure 7-2.

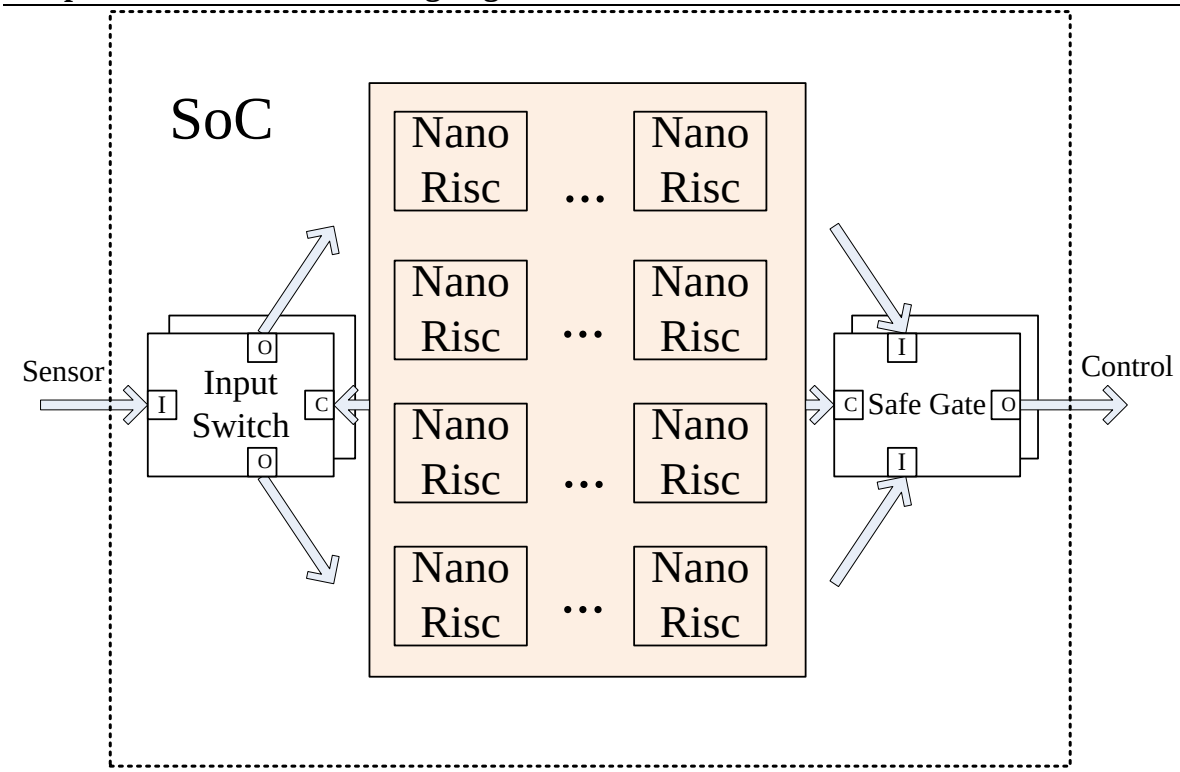

Figure 7-2 Uniform NanoRisc based on the Multicore SoC

7.2.1.2.2. Different cells (4-bit, 8-bit, 16-bit, 32-bit)

In this method, the core processor of the SoC is consisting by several different RISC cores (4-bit, 8-bit, 16-bit and 32-bit). This method eases the implementation and the configuration of SoC. The software can reuse current COST version. However, the power control cell can be up to a 32-bit RISC, so the power efficiency will be not very high. The architecture of this method is shown in the Figure 7-3.

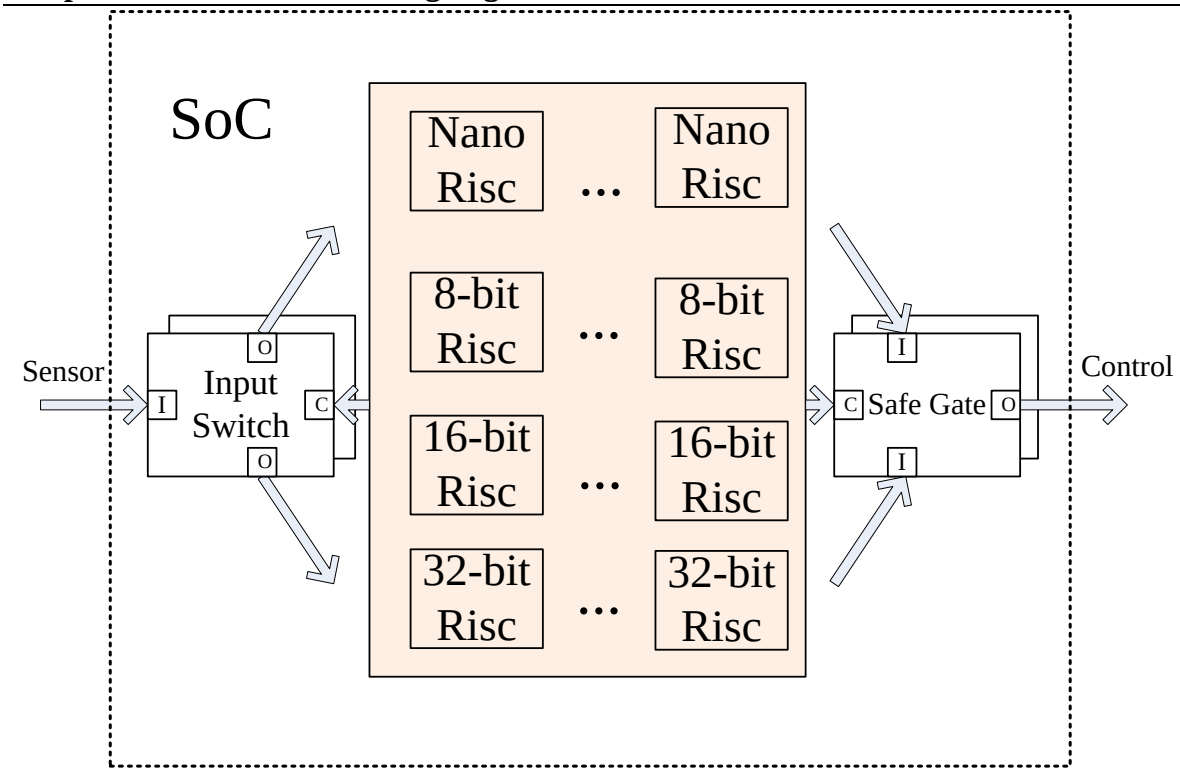

Figure 7-3 Different Common Risc based on the Multicore SoC

7.2.1.2.3. FSMOS Modules

In this method, the core processor of SoC is consisting by an array of special FSMOS modules (Page 124). This method can optimize both the hardware and software of WSN node. In this case, the hardware design will closely combine with the software design. Therefore, this technique will achieve the best power efficiency. The architecture of this method is shown in the Figure 7-4.

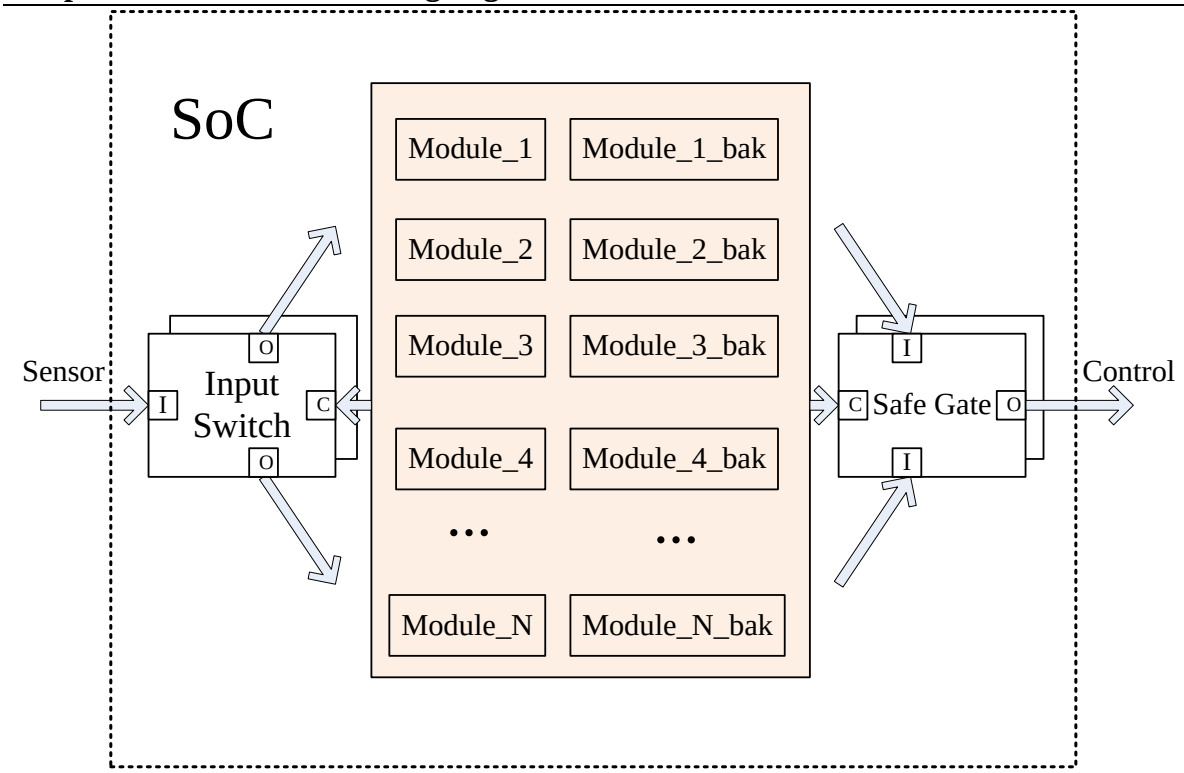

Figure 7-4 FSMOS Modules based the Multicore SoC

7.2.1.2.4. Summary

Table 7-1Table 6-16 provides a summary list of different multicore architectures.

Table 7-1 Different Core Architecture of Multicore SoC

Core Architecture	SoC Implementation	Software Development	Energy Efficient	Generalize	Comment
Uniform NanoRisc Array	Easy	Hard	High	High	
Different cell (4-bit, 8-bit, 16-bit, 32-bit)	Easy	Easy	Low	High	
FSMOS Modules	Hard		Very High	Low	

7.2.1.3. Intra-Chip Multicore Interconnection Networks

The Intra-Chip Multicore Interconnection Networks is the key issue of Multicore SoC. To enable the run-time fault detection and fault recovery, the network has to support HSDTVI. To ease the connection between the different cores, the network should also directly support Multi Point to Point (MP2P) communication, Point to Multi Point (P2MP) communication and Point-to-Point (P2P) communication.

In order to achieve MP2P, P2MP and P2P communication mode, the Intra-Chip Multicore Interconnection Networks needs to be redesigned. The design should consider all the aspect such as speed, memory requirement, energy consuming, connection cost, and robustness.

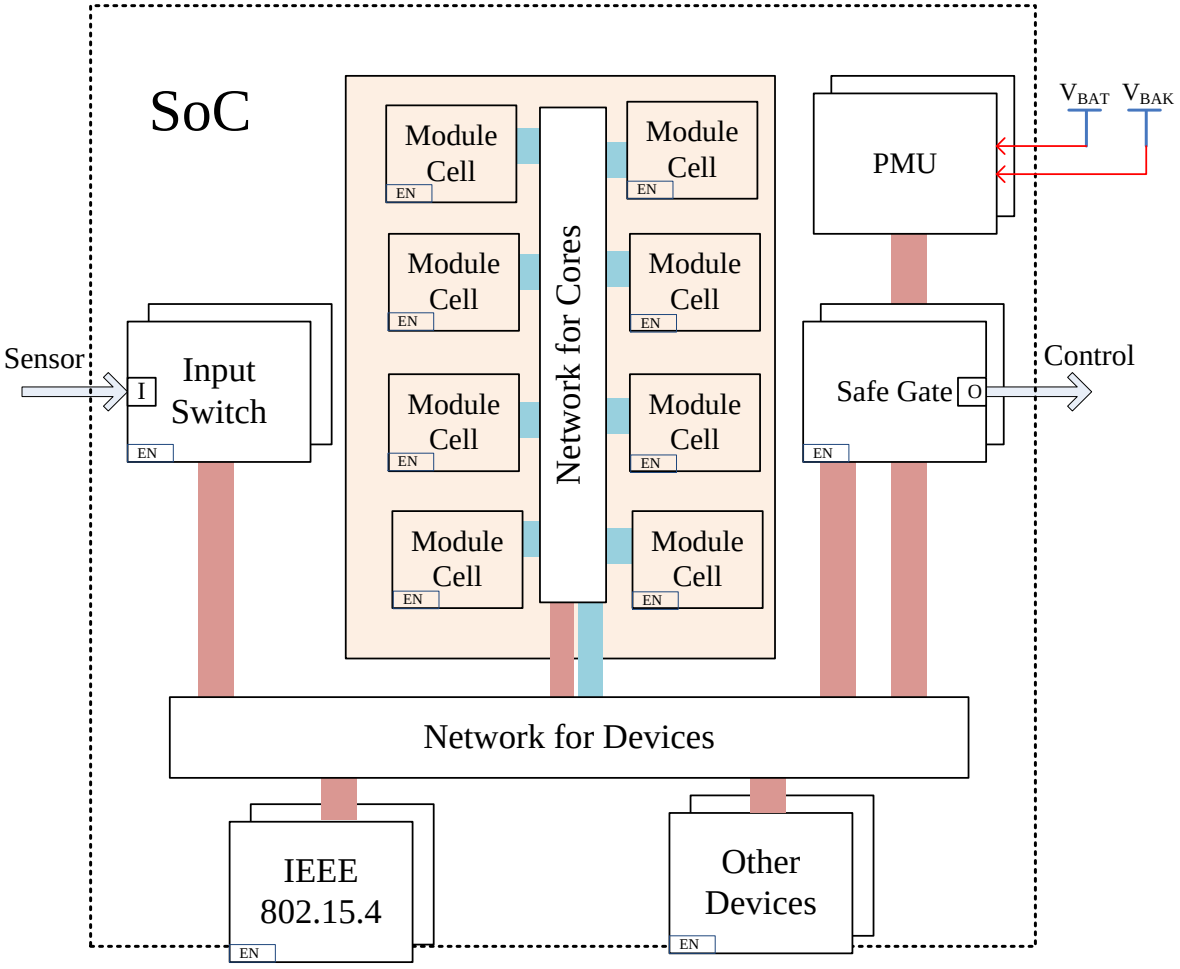

Figure 7-5 Intra-Chip Multicore Interconnection Networks

The Figure 7-5 proposed an Intra-Chip interconnection network with two ranks: Network for Cores (connects all the core modules) and Network for Devices (connects cores with all devices). Besides, the Off-Chip communication of Multicore SoC should also support common interface such as I ₵, SPI, and UART.

7.2.1.4. Integrated Devices

In order to support robust IEEE 802.15.4 connection, the SoC can directly support two independent IEEE 802.15.4 transceivers for high redundancy.

The SoC also needs to integrate all the necessary sensor interface devices for different type of sensors such as GPS on UART port, Decagon soil moisture Sensor on ADC port, and watermark soil moisture Sensor on Timer Port. The device must support smart device running

mode, which means they can run independent and detect event while SoC core node is in sleep mode.

We also need to add special multi HSDTVI devices to enable the run-time fault detection and fault recovery.

7.2.1.5. Constant voltage core and non-constant voltage core

The power supply requirements of different core and different devices are also different. In order to make full use of the high input range of non-constant voltage core that can directly connect to battery, The Multicore SoC should support both constant voltage power input and non-constant power input.

The constant voltage core or modules such as ADC can use the constant power input, while the non-constant voltage core can directly use the battery voltage to go a step further to decrease the power consumption.

7.2.1.6. Summary

The Multicore SoC is specially optimized for WSN sensor node; it can make full use of the advance of multicore architecture. Through black box concept, it can support different requirements with one chip, decrease the Time to Market (TTM), lower the total cost, reuse the technical resource, minimize the development cost for the new project, and ease the stock management. By using SoC approach, we can further decrease the size of the WSN nodes. Higher integration of SoC need less extra components, so it can help to develop lower unit price, smaller form factor and higher reliability WSN node.

7.2.2. Finite State Machine OS: FSMOS

Current existing OS such as Contiki and TinyOS are designed for unicore system. They are not optimized for multicore architecture. Thus, those OS cannot make full use of the advance of multicore architecture. The application needs to handle the inter-communication between different cores from scratch; this will increase complexity and will duplicate work for every application.

In order to ease the implementation and avoid duplicate work, we will develop a native real-time operating system integrated with all the basic functionality related to every aspect of multicore architecture, such as basic hardware driver, standard communication stack, resource management, inter-core communication, power management, and remote process communication. This OS is based on Finite State Machine, so it has been named as Finite State Machine OS (FSMOS). It will be released with user-friendly tools for debugging, test and validation. The section will discuss the concept and some key features of FSMOS.

7.2.2.1. FSMOS Concept

FSMOS is based on following concepts:

- The whole system can be divided into several modules. Each module should have it own Local State. Finite State Machine can describe their running modes.
- FSMOS as a whole big module also has several Global States
- Those Local State and Global States can be easily used in Auto Fault Detection and Fault Recovery
- FSMOS Directly supports Multicore Architecture, provide all the basic multicore system service, such as inter-core communication, multicore power management, and remote process communication.
- FSMOS directly supports IEEE802.15.4, eases the development of WSN application.
- The main source code of FSMOS is C Language, and these codes should directly support cross compiler and can run on different platforms such as AVR, ARM7, ARM11 and PC (WIN and Ubuntu).
- Based on FSMOS, an application can be ported through different platforms without modification.

7.2.2.2. FSMOS High-level Architecture

The high-level architecture of FSMOS is presented on the Figure 7-6. The FSMOS is separated into a number of logical modules each provides a set of APIs accessible for the user.

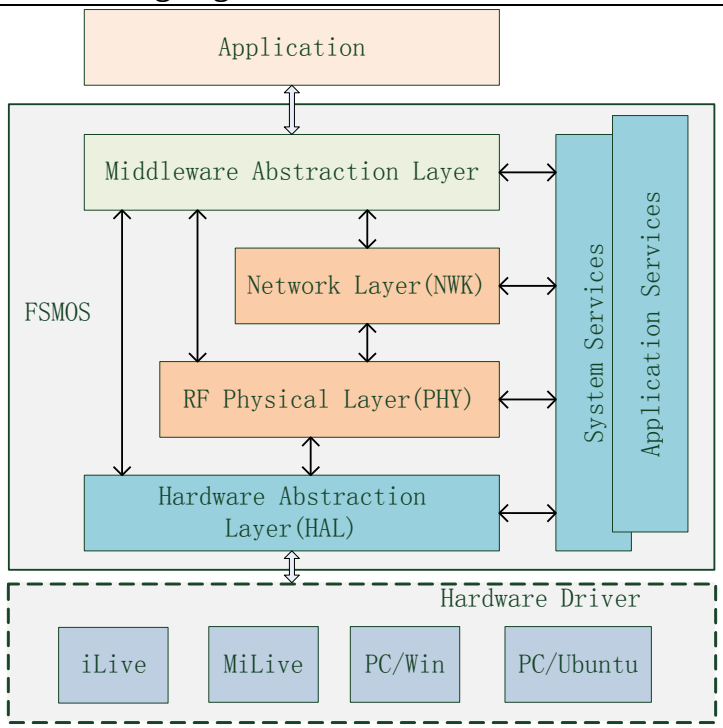

Figure 7-6 Cross Platform of the FSMOS Software Architecture

- Hardware Abstraction Layer (HAL) provides basic hardware dependent platform independent functionality, like hardware timer, sleep control, GPIO access for the radio interface
- Radio physical layer (PHY) provides functions for radio transceiver access. Some of them are accessible only by the network layer (request to send data, data indication); some of them can be used from the application (channel selection, random number generation, energy detection, etc.)
- Network layer (NWK) provides network stack functionality, like Frame Transmission, Frame Reception, and Acknowledgement, Routing, Security, etc.
- System services provide common functions for all layers, which are necessary for normal stack operation. System services include basic types and definitions, software timers, default configuration parameters, encryption module access, etc.
- Application services include modules that are not required by the stack, but are common for most applications, such as Over-The-Air upgrade (OTA), etc.
- Middleware Abstraction Layer provides higher abstraction interface for application development, like node configuration, etc.
- Hardware Driver provides basic hardware dependent platform dependent functionality, like hardware timer, sleep control, GPIO access for the radio interface. The driver included at least AVR, ARM7, ARM11 and PC platform.

7.2.2.3. Remote Modules Based on Remote Processor Call

Debug, test and validate WSN application is still a difficult job. The JTAG tool can help to locate the internal information of WSN node. However, the JTAG tool is a little bit expensive and requires more time for learning how to use it. The tool can only debug one WSN node. If considering the distributed information in a set of WSN nodes, this task will be an even more difficult problem.

There are many methods which have been developed to ease the implementation of WSN application, such as Java Virtual Machine (JVM) (Xing, Xunxing, et al., 2012), Middleware (Xing, Kun Mean, Honglin, & Chengcheng, 2012; Xing, Kun Mean, Hongling, Chengcheng, & Haiying, 2011), etc. but those methods still focus on one single-core node, and real-time debug issue remains uncovered. Here we introduce a FSMOS based debugging method to archive real-time user-friendly debugging experience.

FSMOS is a cross platform design, so it can run on different platform such as AVR, ARM7, ARM11, PC (WIN and Ubuntu), etc. The PC did not have same hardware as WSN mode, in order to enable FSMOS and application over FSMOS running on PC can access real hardware, FSMOS have special modules, remote module, to support hardware accessment.

The Figure 7-7 shows the block diagram of a FSMOS running on PC with remote module. The remote module can provide the same service as real module. The only different between remote module and real module is that remote module can't directly handle the request. So it will forward all the request to a real node in another place, then the request will be handled by that real node. The response will then tranfer back to the remote module in the reverse direction. Then up layer can get the response from remote module simlar as from real module. The request and response transfer and remote execution are all based on Remote Processor Call (RPC) mechanisms in FSMOS.

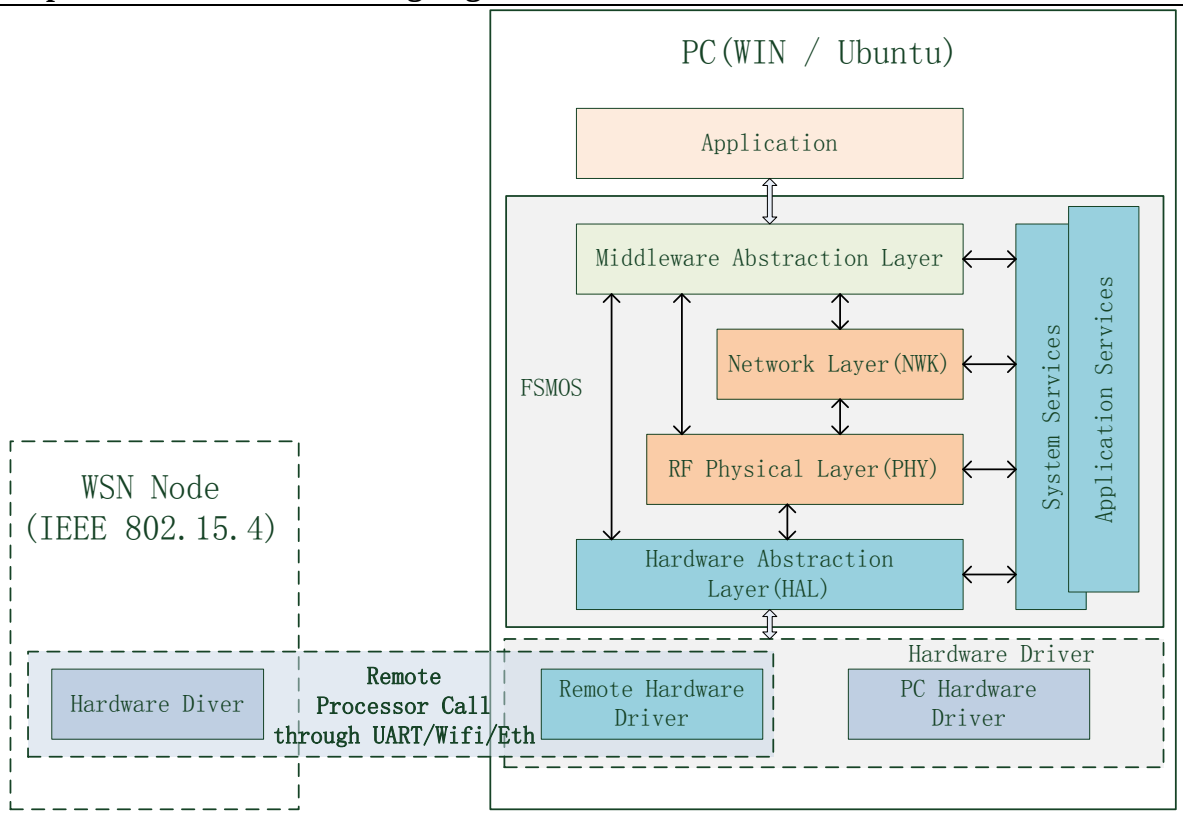

Figure 7-7 FSMOS running on a PC with the Remote Module

Running the FSMOS on a PC can provide many advantages, such as user-friendly IDE, more resource for complex application prototype, more resource for tracing and debugging, etc. And remote modules enable program running on a PC directly access WSN hardware, such as sending and receiving PHY layer packet, controlling GPIO, reading Sensor, etc. Furthermore, gathering distributed information from a set of FSMOS running on a PC is also much easier than directly from WSN nodes.

7.2.2.4. Summary

The FSMOS is optimized for multicore architecture. It can make full use of the advance of multicore architecture. With cross platform and Remote Module, the debugging, testing and validation of WSN application based on FSMOS will be much easier. This also will greatly help to improve the reliability of whole system.

7.3. Perspective

While we have done much work, more work left. In order to provide the entire solution for dependable WSN services, we still have much ongoing work. We think that the existing operating systems such as TinyOS and Contiki are not adapted to multicore WSN node. Thus,

we will implement a Finite State Machine OS (FSMOS), which enables to implement more user-friendly collaborative processing and fault tolerant applications.

In order to ease the implementation of user's application without sacrificing efficiency, we will design and implement an Efficient Context Aware Middleware (ECAM). The middleware can bridge the gap between multiple applications running at application level and FSMOS at system level. In order to archive effective resource utilization, the context aware middleware will provide the entire necessary application interface to control the power states of every core and every component in WSN node. To meet the requirements of differ situation, the context aware middleware will support both knowledge base for static situation and rule-based engine for dynamically changing situation. Furthermore, the context aware middleware supports remote update their rules and knowledge base, which makes applications even more flexible.

Thanks to 6LoWPAN (Y. Chen et al., 2011; Montenegro et al., September 2007), RPL (IETF, 2012) and HTTP, the interoperability of WSN nodes over internet is solved. Therefore, we will follow the IETF standard and work on IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) and Constrained Application Protocol (CoAP)(IETF, 2013) in order to provide our WSN nodes with web service functionalities and to integrate WSNs with the Web seamlessly. We will develop a 6LowPAN-based WSN integrating CoAP, which allows user access WSN data directly from a Web browser. The hardware platform, FSMOS, ECAM, 6LoWPAN and CoAP all together will form the entire solution for dependable WSN services.

We can imagine the applications based on wireless sensor network in the near future. Wireless sensors and control points will present in everywhere and form a lot of WSN. All devices in home or in factory connect to IoT through these invisible wireless sensor networks. These devices all have an IPv6 IP address that user can directly access to them and get any services in anytime from anywhere. There is no cumbersome wiring between these devices any more. The smart devices will interact with the physical world and influence every aspect of our lives.

Nowadays, imagining a world without the Internet is nearly impossible. Do WSNs will have more impacts than Internet for everyday living in the near future? That is an open question.

Bibliography

Actel-Corporation. (2009). IGLOO nano Low-Power Flash FPGAs with Flash*Freeze Technology (pp. 1-120).

Akkaya, Kemal, & Younis, Mohamed. (2005). A survey on routing protocols for wireless sensor networks. *Ad hoc networks, 3*(3), 325-349.

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks, 38*(4), 393-422. doi: 10.1016/S1389-1286(01)00302-4

Al-Karaki, Jamal N, & Kamal, Ahmed E. (2004). Routing techniques in wireless sensor networks: a survey. *Wireless Communications, IEEE, 11*(6), 6-28.

ALD. (2012). Free MTBF Calculator. from http://www.aldservice.com/en/reliability-software/free-mtbf-calculator.html

ARM Ltd. (2013). Cortex-M3 Processor. 2013, from http://www.arm.com/products/processors/cortex-m/cortex-m3.php

Atmel-Corporation. (2011). AT91SAM ARM-based Flash MCU *6175L–ATARM–28-Jul-11* (pp. 1-781).

Atmel-Corporation. (2012a). 8-bit AVRMicrocontroller with Low Power 2.4GHz Transceiver for ZigBee and IEEE 802.15.4 *ATmega128RFA1* (pp. 1-568).

Atmel-Corporation. (2012b). 8-bit Microcontroller with 64K/128K/256K Bytes In-System Programmable Flash *2549O–AVR–05/12* (pp. 1-448).

Atmel-Corporation. (2012c). Reliability Information. from http://www.atmel.com/about/quality/reliability.aspx

Autran, Jean-Luc, Semikh, Sergey, Munteanu, Daniela, Serre, Sébastien, Gasiot, Gilles, & Roche, Philippe. (2012). *Soft-Error Rate of Advanced SRAM Memories: Modeling and Monte Carlo Simulation*.

Avizienis, A., Laprie, J.-C., Randell, B., & Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *Dependable and Secure Computing, IEEE Transactions on, 1*(1), 11-33. doi: 10.1109/TDSC.2004.2

Avizienis, Algirdas, Laprie, Jean-Claude, & Randell, Brian. (2001). Fundamental Concepts of Dependability: Research Report No 1145, LAAS-CNRS.

Baggio, Aline. (2005). *Wireless sensor networks in precision agriculture*. Paper presented at the ACM Workshop on Real-World Wireless Sensor Networks, Stockholm, Sweden.

Baronti, Paolo, Pillai, Prashant, Chook, Vince W. C., Chessa, Stefano, Gotta, Alberto, & Hu, Y. Fun. (2007). Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards. *Computer Communications, 30*(7), 1655-1695. doi: 10.1016/j.comcom.2006.12.020

Basaran, Can, Baydere, Sebnem, Bongiovanni, Giancarlo, Dunkels, Adam, M. Onur Ergin (YTU), Laura Marie Feeney (SICS, editor), et al. (2006). Research Integration:

Platform Survey: Critical evaluation of platforms commonly used in embedded wisents research *IST-004400* (pp. 1-109): Embedded WiSeNts.

Beck, A.C.S., Lisbôa, C.A.L., & Carro, L. (2012). *Adaptable Embedded Systems*: Springer.

Beck, Kent, Beedle, Mike, van Bennekum, Arie, Cockburn, Alistair, Cunningham, Ward, Fowler, Martin, et al. (2001). Manifesto for Agile Software Development *Manifesto for Agile Software Development*.

Bellard, Fabrice. (2005). *QEMU, a fast and portable dynamic translator*. Paper presented at the Proceedings of the annual conference on USENIX Annual Technical Conference, Anaheim, CA.

Benington, Herbert D. (1983). Production of Large Computer Programs. *IEEE Annals of the History of Computing, 5*(4), 350-361. doi: 10.1109/MAHC.1983.10102

Berkeley. (2013). TinyOS, an open source, BSD-licensed operating system designed for low-power wireless devices. 2013, from http://www.tinyos.net/

Blundell, B. (2007). *Computer Hardware*: Thomson.

Boehm, Barry. (1986). A Spiral Model of Software Development and Enhancement. *ACM SIGSOFT Software Engineering Notes, 11*(4), 14-24.

Bokareva, Tatiana. (2013). Mini Hardware Survey. 2013, from http://www.cse.unsw.edu.au/~sensar/hardware/hardware_survey.html

Borkar, Shekhar. (2005). Designing reliable systems from unreliable components: the challenges of transistor variability and degradation. *Micro, IEEE, 25*(6), 10-16.

Broadcom.com. (2013). BCM2835: High Definition 1080p Embedded Multimedia Applications Processor. 2013

Brown, P.L. (1982). *Managing behavior on the job*: Wiley.

BRUMMOND, J. ; , CONGER, S. ; , HART, R. ;, OSBORNE, L. ; , & ZAREAN, M. (2006). Clarus: Concept of Operations.: Federal Highway Administration (FHWA).

Chen, Peter M, Lee, Edward K, Gibson, Garth A, Katz, Randy H, & Patterson, David A. (1994). RAID: High-performance, reliable secondary storage. *ACM Computing Surveys (CSUR), 26*(2), 145-185.

Chen, Yibo, Kun-mean, Hou, Haiying, Zhou, Hong-Ling, Shi, Xing, Liu, Xunxing, Diao, et al. (2011, 23-25 Sept. 2011). *6LoWPAN Stacks: A Survey*. Paper presented at the Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference on.

Dell, Timothy J. (1997). A white paper on the benefits of chipkill-correct ECC for PC server main memory. *IBM Microelectronics Division*, 1-23.

Desai, Uday B., Jain, B. N., & Merchant, S. N. (2010). Wireless Sensor Networks: Technology Roadmap.

DGA. (September 2010). Reliability Methodology for Electronic Systems *FIDES guide 2009 Edition A*.

Dovich, R. A. (1990). *Reliability Statistics*. Wisconsin: ASQ Quality Press.

Dubrova, Elena. (2013). *Fault-Tolerant Design*: Springer.

Bibliography

Feiler, P.H., & Gluch, D.P. (2012). *Model-Based Engineering with AADL: An Introduction to the SAE Architecture Analysis & Design Language*: Pearson Education.

Frankel, David. (2003). *Model driven architecture : applying MDA to enterprise computing*. New York Wiley.

Gao, J., Xu, Y., & Li, X. (2007). Online Distributed Fault Detection of Sensor Measurements. *Tsinghua Science & Technology, 12*, 192-196. doi: 10.1016/s1007-0214(07)70108-6

Haan, Johan den. (2009, 04 February 2009). Roles in Model Driven Engineering. from http://www.theenterprisearchitect.eu/archive/2009/02/04/roles-in-model-driven-engineering

Hewlett-Packard Development Company. (2010). Memory technology evolution: an overview of system memory technologies. from http://h20000.www2.hp.com/bc/docs/support/SupportManual/c00256987/c00256987.pdf

Hewlett-Packard Development Company. (2012). For businesses that run nonstop: HP Integrity NonStop NS2100 Server. from http://www8.hp.com/h20195/v2/GetDocument.aspx?docname=4AA4-2781ENW

Holt, J., Perry, S., Engineering, Institution of, & Technology. (2008). *SysML for Systems Engineering*: Institution of Engineering and Technology.

Hsieh, Hui-Ching, Leu, Jenq-Shiou, & Shih, Wei-Kuan. (2010). A fault-tolerant scheme for an autonomous local wireless sensor network. *Computer Standards & Interfaces, 32*(4), 215-221. doi: 10.1016/j.csi.2009.11.012

IBM. (2011). Virtual I/O Server logical partition. 2013, from http://pic.dhe.ibm.com/infocenter/powersys/v3r1m5/index.jsp?topic=/iphb1/iphb1_vios_virtualioserverpartition.htm

IBM Corp. (2012). Reliability, Availability, and Serviceability Features of the IBM eX5 Portfolio. 2013, from http://www.redbooks.ibm.com/redpapers/pdfs/redp4864.pdf

IEC. (2013). Dependability, Reliability, Maintainability, Maintenance support, International Electrotechnical Commission Technical Committee 56 from http://tc56.iec.ch/index-tc56.html

IETF. (2012). RFC 6550: RPL: IPv6 Routing Protocol for Low power and Lossy Networks.

IETF. (2013). Constrained Application Protocol (CoAP) draft-ietf-core-coap-18.

Intel Corporation. (2008). Dual-Core Intel Itanium Processor 9100 Series., 2013, from http://www.intel.com/content/dam/doc/product-brief/high-performance-computing-itanium-9100-powering-mainframe-solutions-on-flexible-industry-standard-servers-brief.pdf

Intel Corporation. (2013a). Intel® Server Board S1200V3RP Calculated MTBF Estimates from http://download.intel.com/support/motherboards/server/sb/s1200rpcalculatedmtbfestimatesrev1_0.pdf

Intel Corporation. (2013b). PCI Express* Provides Enterprise Reliability, Availability and Serviceability 2013, from http://www.intel.com/content/www/us/en/io/pci-express/pci-express-architecture-devnet-resources.html

Bibliography

International Business Machines Corporation. (1970). *Data Processor*: Data Processing Division, International Business Machines Corp.

J. Polastre, R. Szewcyzk, C. Sharp, & D.Culler. (2004). *The mote revolution: Low power wireless sensor network devices*. Paper presented at the Proceedings of the 16th Symposium on High Performance Chips (HotChips).

Jones, Justin, & Atiquzzaman, Mohammed. (2007). Transport protocols for wireless sensor networks: State-of-the-art and future directions. *International Journal of Distributed Sensor Networks, 3*(1), 119-133.

Kaemarungsi, Kamol. (2012). *Development and Deployment of ZigBee Wireless Sensor Networks for Precision Agriculture in Sugarcane Field*. Paper presented at the APAN 2012 : Asia-Pacific Advanced Network - 33rd Meeting, Chiang-Mai, Thailand.

Kemet. (2012). KEMET FIT Calculator Software. from http://www.kemet.com/page/kemsoft#fit

Keshtgari, Manijeh, & Deljoo, Amene. (2012). A Wireless Sensor Network Solution for Precision Agriculture Based on ZigBee Technology. *Wireless Sensor Network, 4*(1), 25-30. doi: 10.4236/wsn.2012.41004

Khan, Safdar Abbas, Daachi, Boubaker, & Djouani, Karim. (2012). Application of fuzzy inference systems to detection of faults in wireless sensor networks. *Neurocomputing, 94*, 111-120. doi: 10.1016/j.neucom.2012.04.002

Larman, C., & Basili, V. R. (2003). Iterative and incremental developments. a brief history. *Computer, 36*(6), 47-56. doi: 10.1109/MC.2003.1204375

Lavagno, L., Martin, G., & Selic, B.V. (2003). *UML for Real: Design of Embedded Real-Time Systems*: Springer.

Lee, Myeong-Hyeon, & Choi, Yoon-Hwa. (2008). Fault detection of wireless sensor networks. *Computer Communications, 31*(14), 3469-3475. doi: 10.1016/j.comcom.2008.06.014

Lee, W Louis, Datta, Amitava, & Cardell-Oliver, Rachel. (2006). *FlexiMAC: A flexible TDMA-based MAC protocol for fault-tolerant and energy-efficient wireless sensor networks.* Paper presented at the Networks, 2006. ICON'06. 14th IEEE International Conference on.

Levis, P. A. (2006, 10-12 May 2006). *TinyOS: An Open Operating System for Wireless Sensor Networks (Invited Seminar).* Paper presented at the Mobile Data Management, 2006. MDM 2006. 7th International Conference on.

Linear. (2009). Reliability Data. from http://cds.linear.com/docs/Reliability%20Data/r415.pdf

Lipetz, D., & Schwarz, E. (2011, 25-27 July 2011). *Self Checking in Current Floating-Point Units.* Paper presented at the Computer Arithmetic (ARITH), 2011 20th IEEE Symposium on.

Maheshwari, Atul, Burleson, Wayne, & Tessier, Russell. (2004). Trading off transient fault tolerance and power consumption in deep submicron (DSM) VLSI circuits. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, 12*(3), 299-311.

Martin, James. (1991). *Rapid application development*: Macmillan Publishing Co., Inc.

Bibliography

Microsemi-Corporation. (2011). Reliability Report. from http://www.actel.com/documents/ORT_Report.pdf

Microsemi. (2013). Microsemi's Retrieved Nov, 2013, from http://www.microsemi.com/products/fpga-soc/fpga/igloo-nano

Mile Stojčev , Teufik Tokić , Ivan Milentijević. (2004). The Limits of Semiconductor Technology and Oncoming Challenges in Computer Microarchitectures and Architectures. *Ser. Electronics and Energetics, 17*, 285-312.

Minicom Advanced Systems Ltd. (2013). Cost Ramifications of Player Placement in Digital Signage Networks. from www.minicom.com/pdf/PlayerPlacement.pdf

Montenegro, G., Kushalnagar, N., Hui, J., & Culler, D. (September 2007). Transmission of IPv6 Packets over IEEE 802.15.4 Networks *RFC 4944*.

N. Medrano, & S. Celma. (2006). *Wireless sensors for agricultural applications*. Paper presented at the 3èmes Jornadas Hispano Francesas CMC2 - IBERNAM, San Sebastián, Spain.

Nakayama, Hidehisa, Ansari, Nirwan, Jamalipour, Abbas, & Kato, Nei. (2007). Fault-resilient sensing in wireless sensor networks. *Computer Communications, 30*(11-12), 2375-2384. doi: 10.1016/j.comcom.2007.04.023

Nightingale, Edmund B, Douceur, John R, & Orgovan, Vince. (2011). *Cycles, cells and platters: an empirical analysisof hardware failures on a million consumer PCs.* Paper presented at the Proceedings of the sixth conference on Computer systems.

Nowka, Kevin. (2007). Circuits Design for Low Power. 2013, from users.ece.utexas.edu/~adnan/vlsi-07/nowka-low-power-07.ppt

Onsemi. (2012). Reliability Data - Device MTBF/MTTF/FIT. from http://www.onsemi.com/PowerSolutions/reliability.do

Oracle. (2010). Oracle White Paper— Best Practices for Data Reliability with Oracle VM Server for SPARC. 2013, from http://www.oracle.com/technetwork/articles/systems-hardware-architecture/vmsrvrsparc-reliability-163931.pdf

Osterlind, F. (2006). *Cross-Level Sensor Network Simulation with COOJA.*

PandaBoard ES. (2013). PandaBoard ES. 2013, from http://pandaboard.org/content/pandaboard-es

Peng, Jiang, Huijin, Ren, Lei, Zhang, Zhi, Wang, & Anke, Xue. (2006, 0-0 0). *Reliable Application of Wireless Sensor Networks in Industrial Process Control.* Paper presented at the Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on.

Polastre, J., Szewczyk, R., & Culler, D. (2005, 15 April 2005). *Telos: enabling ultra-low power wireless research.* Paper presented at the Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on.

Polastre, Joseph, Szewczyk, Robert, Mainwaring, Alan, Culler, David, & Anderson, John. (2004). Analysis of wireless sensor networks for habitat monitoring. *Wireless sensor networks*, 399-423.

Bibliography

Pressman, Roger S. (2010). *Software Engineering: A Practitioner's Approach*. New York: McGraw Hill Higher Education.

Prolinx Services, Inc. (2013). Agile Software Development. 2013, from http://www.prolinxservices.com/agile.aspx

Raspberry Pi Foundation. (2013). Raspberry Pi. 2013, from http://www.raspberrypi.org/

Reliability Analysis Center. (1996). *Introduction to Software Reliability: A State of the Art Review*: Reliability Analysis Center.

Reliability Information Analysis Center. (2005). *System Reliability Toolkit*: Reliability Information Analysis Center.

Sankarasubramaniam, Yogesh, Akan, Özgür B, & Akyildiz, Ian F. (2003). *ESRT: event-to-sink reliable transport in wireless sensor networks.* Paper presented at the Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing.

SensLAB team. (2013). Very large scale open wireless sensor network testbed. 2013, from http://www.senslab.info/

Spainhower, L., & Gregg, T. A. (1999). IBM S/390 parallel enterprise server G5 fault tolerance: a historical perspective. *IBM J. Res. Dev., 43*(5), 863-873. doi: 10.1147/rd.435.0863

SquareTrade. (2009). Who makes the most reliable laptops? Retrieved 2013, from http://news.cnet.com/8301-17938_105-10400447-1.html

Steve Bostian, Intel Corporation. (2012). Intel® Instruction Replay Technology Detects and Corrects Errors. from http://www.intel.com/content/www/us/en/processors/itanium/itanium-9500-reliability-mission-critical-applications-paper.html

Sutar, Shiv, Jayesh, Swapnita, & Priyanka, Komal. (2012). Irrigation and Fertilizer control for Precision Agriculture using WSN: Energy Efficient Approach. *International Journal of Advances in Computing and Information Researches, 1*(1).

Teodorović, Dušan, & Lučić, Panta. (2006). Intelligent parking systems. *European Journal of Operational Research, 175*(3), 1666-1681. doi: 10.1016/j.ejor.2005.02.033

Texas-Instruments. (2012). CC430 Family User's Guide *SLAU259D* (pp. 1-781).

Texas Instruments Incorporated. (2013). The Hercules™ TMS570 Safety MCU. 2013, from http://www.ti.com/lsds/ti/microcontroller/safety_mcu/tms570_arm_cortex-r4/overview.page

TIK WSN Research Group. (2013). The Sensor Network Museum. from http://www.snm.ethz.ch/Main/HomePage

United Nations. (2013). World Population Prospects: The 2012 Revision. from http://esa.un.org/unpd/wpp/index.htm

US. (1997). Reliability Prediction of Electronic Equipment *Military Handbooks MIL-HDBK-217*.

Wikipedia. (2013). Error detection and correction. 2013, from http://en.wikipedia.org/wiki/Error_detection_and_correction

Bibliography

Xing, Liu, Kun Mean, Hou, Honglin, Shi, & Chengcheng, Guo. (2012, 25-27 June 2012). *Efficient middleware for user-friendly wireless sensor network integrated development environment.* Paper presented at the Wireless Advanced (WiAd), 2012.

Xing, Liu, Kun Mean, Hou, Hongling, Shi, Chengcheng, Guo, & Haiying, Zhou. (2011, 10-12 Oct. 2011). *Efficient and portable reprogramming method for high resource-constraint wireless sensor nodes.* Paper presented at the Wireless and Mobile Computing, Networking and Communications (WiMob), 2011 IEEE 7th International Conference on.

Xing, Liu, Xunxing, Diao, Kun-mean, Hou, Hailun, Zhu, Xin, Liu, Yazhou, Wang, et al. (2012, 4-7 Sept. 2012). *Java Virtual Machine Based Infrastructure for Decent Wireless Sensor Network Development Environment.* Paper presented at the Ubiquitous Intelligence & Computing and 9th International Conference on Autonomic & Trusted Computing (UIC/ATC), 2012 9th International Conference on.

Yan, Lipeng, Chang, Fei, Qin, Weijun, Li, Bo, & Liu, Yan. (2013). Design and Implementation of Testing Platform for Middleware of Wireless Sensor Networks *Advances in Wireless Sensor Networks* (pp. 548-561): Springer.

Yick, Jennifer, Mukherjee, Biswanath, & Ghosal, Dipak. (2008). Wireless sensor network survey. *Computer Networks*, *52*(12), 2292-2330. doi: 10.1016/j.comnet.2008.04.002

Zhang, Bin, & Orshansky, Michael. (2008). *Modeling of NBTI-induced PMOS degradation under arbitrary dynamic temperature variation.* Paper presented at the Quality Electronic Design, 2008. ISQED 2008. 9th International Symposium on.

Zzyzx Peripherals, Inc. (2001). Mean Time Between Failure (MTBF) and Availability – A Primer. 2013, from http://agenda.linearcollider.org/getFile.py/access?subContId=0&contribId=s1t3&resId=0&materialId=0&confId=desya0533

RESUME

Développement d'un capteur multicoeur sans fil à énergie efficient, robuste et modulaire

Le réseau de capteurs sans fil est une technologie clé du 21ème siècle car ses applications sont nombreuses et diverses. Cependant le réseau de capteurs sans fil est un système à très forte contrainte de ressources. En conséquence, les techniques utilisées pour le développement des systèmes embarqués classiques ne peuvent être appliquées. Aujourd'hui les capteurs sans fil ont été réalisés en utilisant une architecture monoprocesseur. Cette approche ne permet pas de réaliser un capteur sans fil robuste et à énergie efficiente pour les applications telles que agriculture de précision (en extérieur) et télémédecine.

Les travaux menés dans le cadre de cette thèse ont pour but de développer une nouvelle approche pour la réalisation d'un capteur sans fil en utilisant une architecture multicoeur pour permettre à la fois d'augmenter sa robustesse et sa durée de vie (minimiser sa consommation énergétique).

Mots-clés— *Capteurs multicœur sans fil ; sensibles au contexte ; tolérance aux pannes ; sûreté ; systèmes distribués ; systèmes embarqué ; Internet des Objets ; Web des Objets.*

ABSTRACT

Development of an Energy Efficient, Robust and Modular Multicore Wireless Sensor Network

The wireless sensor network is a key technology in the 21st century because it has multitude applications and it becomes the new way of interaction between physical environment and computer system. Moreover, the wireless sensor network is a high resource constraint system. Consequently, the techniques used for the development of traditional embedded systems cannot be directly applied. Today wireless sensor nodes were implemented by using only one single processor architecture. This approach does not achieve a robust and efficient energy wireless sensor network for applications such as precision agriculture (outdoor) and telemedicine.

The aim of this thesis is to develop a new approach for the realization of a wireless sensor network node using multicore architecture to enable to increase both its robustness and lifetime (reduce energy consumption).

Index Terms—*Multicore Wireless Sensor Node; Context-aware; Fault Tolerance; Fail-Safe; Distributed Systems; Embedded System; Internet of Things; Web of Things.*