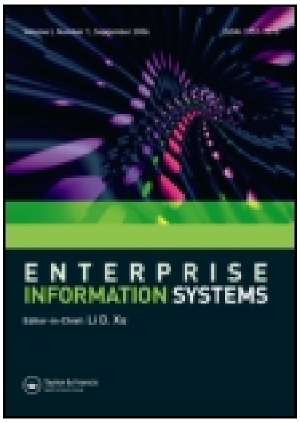

Enterprise Information Systems
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/teis20

Design and implementation of embedded un-interruptible power supply system (EUPSS) for web-based mobile application

De-gan Zhang [a] [b] & Xiao-dan Zhang [c]

[a] Tianjin Key Lab of Intelligent Computing and Novel Software Technology, Tianjin University of Technology, Tianjin, 300191, China

[b] Key Laboratory of Computer Vision and System, Tianjin University of Technology, Ministry of Education, Tianjin, 300191, China

[c] Institute of Scientific and Technical Information of China, Beijing, 100038, China

Published online: 26 Oct 2011.

To cite this article: De-gan Zhang & Xiao-dan Zhang (2012) Design and implementation of embedded un-interruptible power supply system (EUPSS) for web-based mobile application, Enterprise Information Systems, 6:4, 473-489, DOI: 10.1080/17517575.2011.626872

To link to this article: http://dx.doi.org/10.1080/17517575.2011.626872

Enterprise Information Systems
Vol. 6, No. 4, November 2012, 473–489

Design and implementation of embedded un-interruptible power supply system (EUPSS) for web-based mobile application

De-gan Zhang[a,b] and Xiao-dan Zhang[c]*

[a]*Tianjin Key Lab of Intelligent Computing and Novel Software Technology, Tianjin University of Technology, Tianjin 300191, China;* [b]*Key Laboratory of Computer Vision and System, Tianjin University of Technology, Ministry of Education, Tianjin 300191, China;* [c]*Institute of Scientific and Technical Information of China, Beijing 100038, China*

(*Received 21 December 2010; final version received 3 August 2011*)

With the growth of the amount of information manipulated by embedded application systems, which are embedded into devices and offer access to the devices on the internet, the requirements of saving the information systemically is necessary so as to fulfil access from the client and the local processing more efficiently. For supporting mobile applications, a design and implementation solution of embedded un-interruptible power supply (UPS) system (in brief, EUPSS) is brought forward for long-distance monitoring and controlling of UPS based on Web. The implementation of system is based on ATmega161, RTL8019AS and Arm chips with TCP/IP protocol suite for communication. In the embedded UPS system, an embedded file system is designed and implemented which saves the data and index information on a serial EEPROM chip in a structured way and communicates with a microcontroller unit through I^2C bus. By embedding the file system into UPS system or other information appliances, users can access and manipulate local data on the web client side. Embedded file system on chips will play a major role in the growth of IP networking. Based on our experiment tests, the mobile users can easily monitor and control UPS in different places of long-distance. The performance of EUPSS has satisfied the requirements of all kinds of Web-based mobile applications.

Keywords: EUPSS; Web; SEEPROM; FAT; monitoring and controlling; mobile application

1. Introduction

With the rapid development of Web-based mobile applications, people not only hope to obtain data from it, but also hope to access behaviour through it. Access demands will increase as a direct result of Metcalf's Law, which states that bandwidth capacity doubles every 12 months. Under this requirement, embedded internet (EI) technology grows rapidly. EI systems place new demands on IP networking and embedded web-server resources (Satyanarayanan 2001). But as we know, it will most surely be driven by the embedded systems industry. Increased access requires new levels of performance and reliability from the embedded file system on the web server side. The structure is required to continuously support higher connection levels and

*Corresponding author. Email: gandegande@126.com; zhangshenyang@126.com

ISSN 1751-7575 print/ISSN 1751-7583 online
© 2012 Taylor & Francis
http://dx.doi.org/10.1080/17517575.2011.626872
http://www.tandfonline.com

intensive data processing using conventional file systems, and can seem like a luxury to many IT professionals, even impossible in some environments (Zhang 2009, Xu *et al*. 2011).

Embedded system is an application-oriented computer system; its hardware and software can be customised (Lee 2010, Wolf 2010). It mainly includes microprocessor, peripheral units, embedded operating system and application software. Embedded operating system generally includes hardware drivers, OS kernel (including CPU management, memory management, IO management, interrupt management, file management, etc.), network protocol suite, graphical user interface (GUI), Web browser, etc.

As one of the embedded systems, embedded un-interruptible power supply (UPS) system (in brief, EUPSS) can be designed and developed for Web-based mobile application. During mobile application, it can not only monitor the status of UPS, but can also control it remotely through the internet (Web-based network) (Deborah and Ramesh 2008, Hu 2009).

In the EUPSS, in-SEEPROM file systems based on inter-IC Control (I^2C) bus (ISEROMFS) provide an embedded file system solution for reducing the cost of performance and scalable in mission-critical web or other dedicated servers. ISEROMFS has low cost and high reliability by using SEEPROM which eliminates complex layout and instead connects to a microcontroller unit (MCU) only with two pins.

By structuring data access to file system instead of unsystematic non-volatile memory area, very substantial performance improvements can be achieved over a conventional data access. Even when they are all formatted in non-volatile memory, data must still be read and written by specific operation on the memory instead of the conventional way.

In this article, a design and implementation solution of embedded UPS is brought forward for long-distance monitoring and controlling of UPS in Web-based mobile applications.

2. Relative works

EI technology will be applied in many fields, such as family appliance and industry controls. EI technology can make electronic equipment intelligent and bring great convenience to the user by obtaining information and controlling. Based on our review of many references, we have known the existing works in this domain, especially the papers published during recent years, such as the references from 2007 to 2010 (Lavagno *et al*. 2007, Gong *et al*. 2010). As we know, research has been done on the designing of internet embedded systems, such as embedded printer, embedded air-condition, embedded other appliances, and so on. Some research focus on hardware–software co-design of embedded systems, such as embedded arm-based mobile robot. Some research has been done on real-time embedded systems, such as embedded TV, PDA, and so on. After analysing under the banner of the internet of things (IOT), we know the limitations of these existing systems. There is no effective file system in these systems, but at the same time, the functions supporting different kinds of mobile applications, such as B/S application and peer to peer application in mobile process for pervasive services are very weak.

Single chip and MCU have been widely used in domestically and in industries. But the chip and MCU are embedded in special equipments and so they have their own shells. Thus, they are named as embedded systems. At present, all embedded systems are at the state of independency applications. In all the embedded systems, MCU is the major component. With the help of other equipment, it can achieve special function. In some industrial applications or automobile manufacturing, in order to achieve communication among multi-MCUs, network technologies such as CAN, RS-232, RS-485, and so on are used. But all these network technologies are too limited compared with the internet technology and their efficient radius is short. The chief reason is there are much fewer communication protocols to be used in network and isolated from internet. Now internet has become the most important basic establishment and the main channel to exchange information. If embedded system can combine with internet then we can transmit message to any corner of the world (Atmel 2009a–e).

The attempt to link the embedded system and the internet has a long history, but the main difficulty is that the capability of computer memory and computing speed to perform communication protocol cannot meet the resource need of internet application. The resource lacked seriously in embedded system which mainly uses 8-bit or 16-bit microcontrollers, and only a few applications use 32 bit. In order to support TCP/IP protocol suite, a mass of resource will be used. For a long time, we have not been able to perform the mission of linking computers and equipment. Today we combine the intelligence of computer and essential element of equipment to reconstruct an EI system, which can be embedded into equipment that has isolated network intelligence. This is a significant process of the existing of EI.

The main difference between the common embedded system applications and EI applications is communication; communication is better in EI applications. EI systems transmit control message through internet, which is the main feature that distinguishes them from the common embedded systems. EI software includes two sections: first, a web server function should completed in equipment part, in this part program usually stored in Flash ROM of embedded chip, beside this there are also network protocol that used in application. Second, browser as a good client program should be used in user terminal, besides browser there are also some special client applications. A popular model is browser and web server architecture (B/S). The advantage of using browser is its convenience and popularity – nowadays every PC has a browser installed. Through browser we can access the equipment everywhere.

As we know, there is no real embedded UPS for Web-based mobile application, so we present its design and implementation process. Especially, we present an embedded file system, which saves the data and index information on a serial EEPROM chip in a structured way and communicates with MCU through I^2C bus. This file system can be embedded into embedded UPS and other new information applications. Designers can access and manipulate local data on the web client side. Of course, the potential applications of ISEROMFS include most aspects of traditional IT as well as many new requirements and other services that previously could not be delivered in a cost-effective manner.

3. Hardware structure of EUPSS

The microcontroller used in the embedded UPS system is ATmega161, which is the product of America Atmel Corporation. The ATmega161 is a high-performance, low-power CMOS 8-bit microcontroller based on the AVR RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega161 achieves throughputs approaching 1 MIPS per MHz. It is widely applied in radar system, robot fields, etc.

The ATmega161 has the following features: 16K bytes of In-System-programmable Flash, 512 bytes EEPROM, 1K bytes SRAM, 35 general purpose I/O lines, Real Time Counter, three flexible timer/counters, internal and external interrupts, two programmable serial UARTs, programmable watchdog timer with internal oscillator, a SPI serial port.

NIC (Network Interface Controller) used in this UPS system is a RTL8019AS which is the product of ARM and REALTEKSEMI Corporation. It is in charge of the data transmission between ATmega161 and transport medium.

The RTL8019AS has features as follows.

The RTL8019AS is a highly integrated Ethernet controller which offers a simple solution to implement a Plug and Play NE2000 compatible adapter with full-duplex features. The full-duplex function enables simultaneously transmission and reception on the twisted-pair link to a full-duplex Ethernet switching hub. The RTL8019AS supports not only plug and play function but also jumper and proprietary jumperless options.

In order to offer a full plug and playsolution, the RTL8019AS provides the auto-detect capability between the integrated 10BaseT transceiver, BNC and AUI interface. Furthermore, it supports eight IRQ lines and 16 I/O base address options.

The RTL8019AS is built in 16K-byte SRAM of a single chip. It is designed not only to provide more friendly functions but also to save the effort of SRAM sourcing and inventory.

Hardware in the embedded UPS system includes AVR ATmega161 microcontroller, RTL8019AS NIC, 512-bytes EEPROM, 8K bytes SRAM, 528K bytes AT45DB041B Data Flash, HT1380 Serial Timekeeper Chip, UART0 and UART1, etc. Figure 1 is the hardware structure.

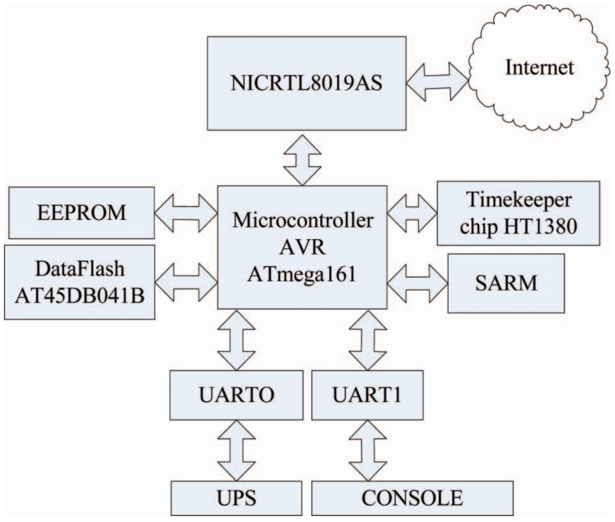

Figure 1. Hardware structure of the embedded UPS system.

Now we introduce the hardware and their functions as follows.

(1) NIC:
  (a) Receives packages sent to the system by remote hosts through internet, puts them to SRAM waiting for processing by AVR microcontroller.
  (b) Sends data packages to transport medium correctly.
(2) AVR microcontroller:
  (a) Controls NIC to receive and send data packages.
  (b) Processes the data packages received by NIC.
  (c) Send some request data packages, such as ARP and DNS request packages.
  (d) Maintains the clock chip.
  (e) Operates file system.
  (f) Receives commands sent by the CONSOLE to configure the file system.
  (g) Provides status and log of UPS, or controls the UPS.
  (h) Sends warning message for the events of UPS to a few of email addresses through SMTP.
  (i) Supports SNMP to response for the standard network management.
(3) EEPROM: This is mainly used to store some variables, including 'OEM Name', 'Product Model', 'Language ID', 'Gateway IP', 'Subnet Mask', 'DNS IP', 'Local IP', 'Local MAC', 'Time Server IP', 'SMTP Server Name', 'SNMP Trap IP', 'Email Address', 'Authentication Control Block Array' and 'HTTP Port'.
(4) SRAM: This is used to store some variables and buffers, including 'File System Type', 'File System Upgrade Flag', 'TCP Connect Control Block Array', 'File Operation Control Block Array', 'Authentication Control Block Array', 'Email Sent State', 'Package Receive/Send Buffer', 'CONSOLE Receive/Send Buffer', 'SNMP MIB', 'UPS Options List', 'Default Web Page Filename' and 'Temp Variables Buffer'.
(5) Data flash: This is used to store some information, including 'File System', 'UPS Data Log', 'UPS Event Log', 'UPS Day Arrange' and 'UPS Week Arrange'.
(6) Controls the UPS through UART0.
(7) Configures the system through UART1 or network.
(8) The HT1380 Timekeeper Chip is used to maintain this system clock.

4. Hardware of ISEROMFS

The hardware architecture of ISEROMFS made by Arm Corporation in EUPSS is showed in Figure 2. The I^2C bus is a two-wire synchronous serial interface consisting of one data (SDA) and one clock (serial clock line (SCL)) line. By using open drain/collector outputs, the I^2C bus supports any fabrication process (CMOS, bipolar and more). The SEEPROM and the MCU need two wires to connect to I^2C bus and they can communicate with each other.

The I^2C bus is a multi-master bus where one or more devices, capable of taking control of the bus, can be connected. Because there is only one master connected to the bus in ISEROMFS, it does not need to support handling of bus contentions and

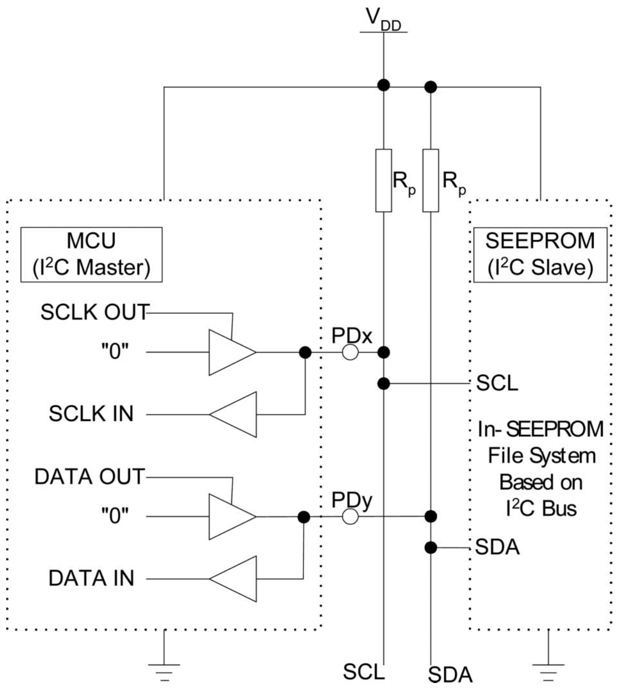

Figure 2. Hardware structure of ISEROMFS.

internal master access (a master accessing another master). Only master devices can drive both the SCL and SDA lines, while a slave device is only allowed to issue the data on the SDA line (Atmel 2009a–e).

Data transfer is always initiated by a bus master device. It is defined to be a START condition that high to low transition is done on the SDA and SCL line. A START condition is always followed by the 7-bit slave address and then by a data direction bit. The slave device acknowledges to the master by holding SDA low for one clock cycle. If the master does not receive any acknowledgement, then the transfer will be terminated. Depending on the data direction bit, the master or slave transmits 8-bit of data on the SDA line. The receiving device then acknowledges the data. Multiple bytes can be transferred in one direction before a repeated START or a STOP condition is issued by the master. The transfer is terminated when the master issues a STOP condition. A STOP condition is defined by a low to high transition on the SDA line when the SCL is high.

Both I^2C lines (SDA and SCL) are bi-directional. Therefore, outputs must be of the open-drain or an open-collector type. Each line must be connected to the supply voltage via a pull-up resistor. A line is logic high when none of the connected devices drives the line, and logic low if one or more devices drive the line low.

5. Software structure of EUPSS

Figure 3 is the software structure of the embedded UPS system. It includes three layers: drivers layer, TCP/IP protocol suite and file system layer, and tasks layer.

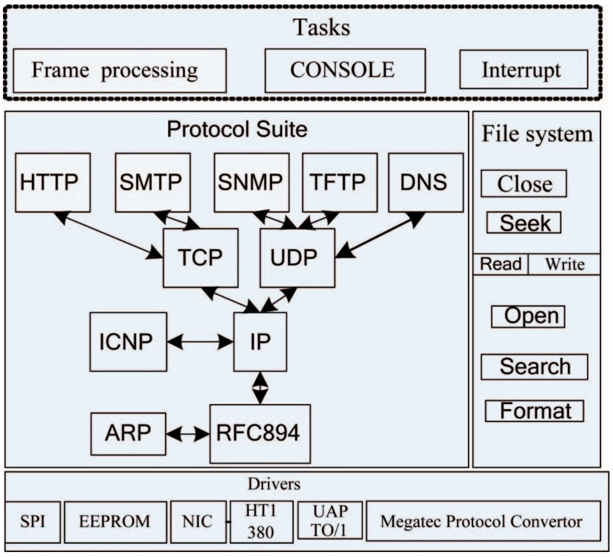

Figure 3. Software structure of the embedded UPS system.

The hardware and their functions implemented by us are as follows.

(1) Drivers:
  (a) Initialise, read and write data flash through SPI.
  (b) Read and write EEPROM.
  (c) Control RTL8019AS to receive and send packages using 'Buffer Loop' technology to improve capability.
  (d) Control HT1380 timekeeper chip.
  (e) Control UART0/1 to read and write.
  (f) Convert TCP/IP protocols to Megatec Protocol which is used by the UPS.
(2) TCP/IP Protocol Suite:
  (a) ARP protocol module provides local MAC.
  (b) ICMP protocol module provides the function of testing network connectivity.
  (c) TCP and UDP protocols module provide connection-oriented service and connectionless service for the higher layer.
  (d) HTTP protocol module provides www service of web server.
  (e) TFTP protocol module provides the upload function. User can upload the web pages and java applets to the data flash to control the UPS remotely.
  (f) DNS module can convert domain name (such as SMTP server, time server) to IP address.
  (g) SMTP protocol module can send some warning messages for UPS to specified email address.
  (h) SNMP protocol module provides a standard UPS network management interface. User can use OpenView, SNMPView, etc. software to monitor and control UPS. At present, it only supports SNMPv1.

(3) Tasks:
  (a) Frame processing: This includes some subtasks, such as 'provides local MAC', 'responses for the ping request', 'gets IP from domain name', 'uploads file system', 'downloads web pages', 'monitors and controls UPS', 'configuration options of the system', 'provides email warning and SNMP standard network management'.
  (b) Console management: It provides the function of reading and upgrading options of the system.
  (c) ISR (Interrupt service routine): It provides the function of refreshing status of the UPS, recording UPS data log, maintaining four timer used by per TCP connection (retransmit timer, etc.) and TTL (time to live) for file open time.

6. Logic structure of EUPSS

Figure 4 is the logic structure of EUPSS. Firstly, user configures options (such as IP, MAC) of the system using toolbox client application, and uploads the web pages and java applets which are designed by user or provided by the system.

Secondly, user accesses this system through a standard browser remotely, and monitors the UPS through java applets.

Thirdly, user can also control the system using specified client application.

Finally, user can use common UPS network management software to operate the UPS for general function. Authentication is needed for advanced function.

7. Software structure of ISEROMFS

The software structure of ISEROMFS in embedded UPS system consists of data structure of file system and operations on the data structure. For simplicity, directory structure is only one level, that is to say, all the files are in one directory. The most important part of file system structure is file allocation table (FAT). To simplify the task for embedded developers, a set of embedded file system access functions, based on the low-level functions of I^2C bus, is discussed.

The file system in embedded UPS system is designed for storing web pages, pictures and data files together with configuration files, as discussed in Atmel (2009a–e). These files normally do not need frequent updates. A file system with FAT in a SEEPROM chip means efficient access to it. FAT of ISEROMFS, showed in Figure 5, contains all the necessary information to read, write or delete files in the file system.

In embedded UPS environment, the size of an embedded file system is usually 16K–2048K because of the limitation of resource. In order to port it easier, we enlarge FAT moderately. The first 4-byte field is recorded total size of the file system in bytes, so the maximum size can reach $2^{32} = 4$G bytes, which meets maximum volume requirements in embedded UPS system. Available size can be read from next 4-byte field. The total number of files and available files in this file system locate in next two 2-byte fields, respectively. Following the 12-byte summary information are file records, each of which occupies 24 bytes to specify file name, address in memory area and size. Each file name is a null-terminated C string, whose maximum length is 15 bytes in a 16-byte field. It takes two 4-byte for Address and size of each file, respectively.

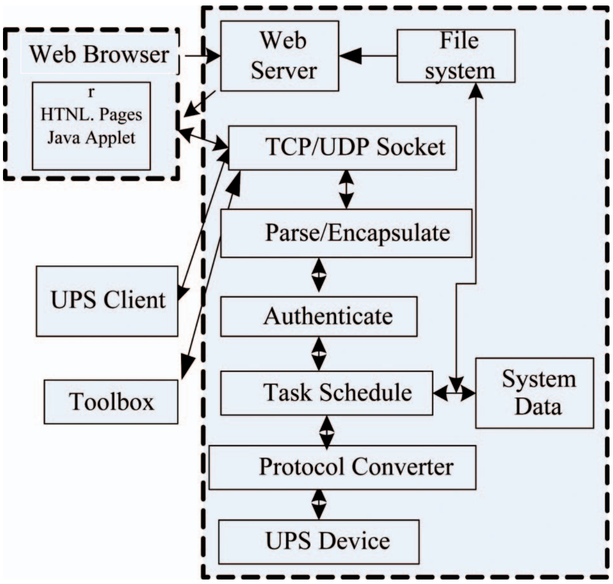

Figure 4. Logic structure of the EUPSS.

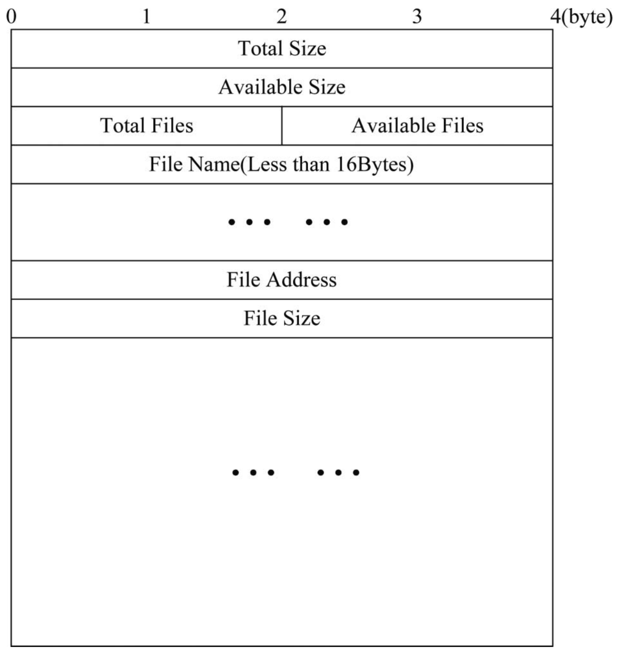

Figure 5. FAT of ISEROMFS.

The volume of all transferring on I^2C bus is byte size. Each byte is followed by an acknowledge bit set by the receiver. The slave address byte contains both a 7-bit address and a read/write bit. Change of data on the SDA line is only allowed during the low period of SCL. This is a direct consequence of the definition of the START and STOP conditions. A more detailed description and timing specifications can be found in the references (Zhang 2006, 2008, Demian 2010).

Without the loss of generality, we discuss I^2C bus low-level functions using a kind of MUC which has one digital I/O port, like port B, at least supplying two pins for I^2C bus. The only resource used by the I^2C master functions presented is the two pins for SCL and SDA on port B. Since the I^2C bus is synchronous, the duty cycle and the period time to the SCL is not critical. Therefore, it is unnecessary to 'fine-tune' the functions, which would lead to an increase of the program size.

There are two types of delays used in this implementation: quarter period and half period delay. For I^2C in normal mode (200 kHz), these delay must be $t_{\text{quarter}} > 1.0\ \mu s$ and $t_{\text{half}} > 2.8\ \mu s$. Because I^2C is in fast mode (800 kHz), the parameters are $t_{\text{quarter}} > 0.3\ \mu s$ and $t_{\text{half}} > 0.7\ \mu s$. There are a large number of possible implementations of the delay loop (Zhang 2008). All the implementations depend on the MCU clock frequency. It is impossible to make a generalised version which is efficient for all clock speed.

The following are the I^2C low-level functions.

(1) I2C_Init: Initialise SCL and SDA lines. The SCLP and SDAP constants located at the top of program code chose the pin number on port B. It is possible to use any pin on any port by changing the program code if required.
(2) I2C_Start: Generate start condition and send slave address. All data transfer must start with this subroutine. When a transfer is done, the 'I2C_Stop' must be called. When the bus is free (after 'I2C_Stop' is called), all registers are free for other usage.
(3) I2C_RepStart: Generate repeated start condition and send slave address. A repeated START can only be given after a byte has been read or written.
(4) I2C_Write: Write data (one byte) to the I^2C bus. This function is also used for sending the address.
(5) I2C_GetAck: Get slave acknowledge response. The reason for separating this subroutine from the 'I2C_Write' routine is to get a more readable program code.
(6) I2C_Read: Read data (one byte) from the I^2C bus.
(7) I2C_PutAck: Put an acknowledge bit, which depends on carry flag (whether set or not). Separating this subroutine from the 'I2C_Read' routine is convenient for the user if an acknowledgement is based on the result of the read operation.
(8) I2C_Stop: Generate stop condition. When a transfer is done, the 'I2C_Stop' must be called. When the bus is free (after 'I2C_Stop' is called), all registers are free for use.
(9) I2C_DoTransfer: This routine is implemented only for convenience. It uses the direction bit from the last sent address byte, to decide whether to call the 'I2C_Read' or the 'I2C_Write' routine.

A file system with a FAT in one part of the EEPROM is necessary in some case since this would lead to an efficient access to a specified file or files. The file system supports traditional MS-DOS file names with eight plus three characters, and it is case sensitive. In most EI environments, efficient reading is very important for an overall good performance, which is congruent to use SEEPROM. The number of concurrently opened files is limited by FILE_MAX_OPEN_FILES defined in file.h. Only one file can be opened for writing at a time.

The access functions for ISEROMFS are as follows:

(1) char fileOpen (char **filename*): Open a file whose name is specified by string – *filename*. If the file exists in file system, this function returns the number of file control block (FCB) in SRAM which records relevant information for accessing the file, returns *0xFF* otherwise.
(2) char fileSeek (char *nFCB*, int *offset*, char *nWhere*): Set the file position indicator for the file pointed by *nFCB*. The new position measured in bytes is obtained by adding offset bytes to the position specified by *nWhere*. If *nWhere* is set to SEEK_SET, SEEK_CUR, or SEEK_END, the *offset* is relative to the start of the file, the current position indicator, or end-of-file, respectively. A successful call to the fileSeek function returns *0x00*, returns *0xFF* otherwise.
(3) char fileRead (char *nFCB*, char **buf*, char *nLen*): Read *nLen* bytes of data from the file pointed by *nFCB*, storing them at the location given by *buf*. If the remainder length of the file is less than *nLen*, this function only reads the remainder data. The return value is the number of bytes read from the file really.
(4) char fileCreate (char **filename*): If the file which is specified by *filename* does not exist, it creates a file and returns the FCB number, returns *0xFF* otherwise.
(5) char fileWrite (char *nFCB*, char **buf*, char *nLen*): Write *nLen* bytes of data to the file pointed by *nFCB*, obtaining them from the location given by *buf*.
(6) char fileDelete (char **filename*): If the file specified by *filename* exists, it will be deleted and returns *0x00*, returns *0xFF* otherwise. In reality, the first two bytes of the file name in FAT are marked by two bytes 0x00 and 0x45. Until the function fsClearUp is called, the deletion is completed.
(7) char fileClose (char *nFCB*): If a successful call of the function is done, it frees the FCB identified by nFCB and returns 0x00, returns 0xFF otherwise.
(8) void fsClearUp (void): Clear up the file system.

8. Experiment and test

An embedded UPS system is built based on AVR ATmega161 microcontroller, RTL8019AS NIC and Arm chips. It is suitable for embedded system and is of low cost. In our projects (Zhang 2008, 2009), we have done many experiments and tests. At the same time, we have done other tests and comparisons for different Web-based mobile applications with other methods introduced by Fisher (1999) and Mahler (2007). Based on our results, we conclude that this system provides the following features: (1) provides 10BaseT interface; (2) supports ARP, IP, ICMP, TCP, UDP, TFTP, HTTP, DNS, SMTP, SNMP, etc.; (3) supports UPS protocol – Megatec Protocol; (4) customs web pages and java applet; (5) monitors UPS using java applets or special client software through C/S model; (6) reacts quickly (<2 ms); (7) provides in-system-programmable capability. MCU and SEEPROM of the embedded UPS system for experiment are Atmel AT90S8515 and Atmel AT24C256, respectively. It takes less than 2 s for low-level functions to read 32K bytes, that is, reading one byte from SEEPROM occupies about 61 μs. Writing one byte takes almost 10 ms. The mean time of opening a file is about 20 ms, which includes finding the file and writing its FCB.

In one of our experiments, firstly, we adopted fuzzy-neural network fusion (FNF)-neural method to deal with relative data during we test our designed and implemented EUPSS (Mori *et al.* 1998, Paul and Richard 2006, Reid 2007, Saha and Chang 2007). In order to express conveniently, we consider the data from an embedded UPS system. The data can be analysed by fuzzy data curve: We can evaluate the influence to output by input variable. x_i ($i = 1,2, \ldots ,n$) expresses the input variable data, and o expresses the output. Suppose there are m training sample points, x_{ik} ($i = 1,2, \ldots ,n,\ k = 1,2, \ldots ,m$) is sample point of the kth sample connecting the ith input variable. For each input variable x_i, the definition of variable fuzzy membership degree function is:

$$y_{ik}(x_i) = 1.0 - \exp\left(-(a/(x_{ik} - x_i))^{4.0}\right) \quad (k = 1, 2, ..., m) \tag{1}$$

where y_{ik} is membership degree function of input x_i to sample point k. a is about 5% of least gap of x_i. Each pair (y_{ik}, $o^{(k)}$) has the relative fuzzy rule for x_i, i.e. IF x_i is y_{ik} (x_i), THEN o is $o^{(k)}$. For each input variable x_i, with the following equation, we can obtain a fuzzy data curve r_i:

$$r_i(x_i) = \sum_{k=1}^{m} y_{ik}(x_i) o^{(k)} \Bigg/ \sum_{k=1}^{m} y_{ik}(x_i) \tag{2}$$

In Web-based mobile application environment, the input signal data of the embedded UPS system is slow. A slight tuning can be done with online training in fuzzy-neural network, so the fuzzy-neural network must be adaptive (D'Mello and Ananthanarayana 2010, Capozucca and Guelfi 2010). As we know, the change speed of making decision fusion result for embedded UPS system is faster and much more than the input signal variable or data, so according to the change rate of input data, we design a function as the input. The expectation can be got by fusion result and this function, and the expectation can be used as training sample to do online slight tuning (Koch and Mitlohner 2010, Wickramasinghe and Gunawardena 2010). Suppose the belief degrees of input data are, $\mu_1, \mu_2, ..., \mu_n$ the constructed function may be selected as follows:

$$f(\mu_1, \mu_2, ..., \mu_n) = \begin{cases} +0.01, & \sum_{i=1}^{n} \mu_i(k) - \sum_{i=1}^{n} \mu_i(k-1) \succ 0 \\ -0.01, & \sum_{i=1}^{n} \mu_i(k) - \sum_{i=1}^{n} \mu_i(k-1) \leq 0 \end{cases} \tag{3}$$

where $\mu_i(k)$, $\mu_i(k-1)$, is the kth and the $(k-1)$th belief degree of ith input signal data, respectively. The expectation value of slight tuning is $y + f$ during each fusion, y is the fusion result.

In the test case (Table 1) of mobile Web-based applications, based on the approaches of fuzzy-neural network mentioned above (Julio 2008, Mahavir 2008, Kakousis *et al*. 2010) for embedded UPS system, we got the following statistical results.

When the number of FNF-neural cell is 10, the average error is as follows:

$$M_z = \frac{1}{10} \times \sum_{i=1}^{10} |Y_i - Z_i| = \frac{28.5}{10} = 2.85$$

The average error rate is as follows:

$$\left(1 - \frac{M_z}{\frac{1}{10} \times \sum_{i=1}^{10} Y_i}\right) \times 100\% = 1 - \frac{2.85}{72.5} \times 100\% = 3.93\%$$

Similarly, when the number of FNF-neural cell is 8, the average error rate is 4.12%.

The relationship between the number of FNF-neural cell and convergence error is shown in Figure 6.

From Figure 6, we can judge that the belief degree result of the embedded UPS system is high, which is close to 1. At the same time, we find that the average error ratio is small, which is lesser than 4%. After analysing our experiment tests, we conclude that the performance of embedded UPS system meets the request of all kinds of Web-based mobile applications.

Now we give the test results by other relative methods (Reid 2007, Saha and Chang 2007, Shen and Chou 2010) in the same experimental examples. Firstly, we use the Extended D-S method (EDS) and classical D-S method to compare the belief degree result of the embedded UPS system. Then we use the Extended D-S (Zhang

Table 1. Relationship between neural cell and error rate.

Name	Neural cell	Sample dataset	Trained number	Average error rate (%)
FNF	8	2 KB	2000	4.12
FNF	10	2 KB	2000	3.93

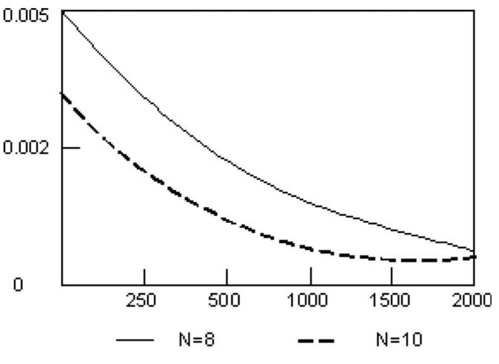

Figure 6. Relationship between the number of FNF-neural cells and convergence error.

2008, 2009), RST and BT to compare the belief degree result of the embedded UPS system (Duan *et al*. 2011, Narman *et al*. 2011).

With the increase of decision evidences made by the embedded UPS system, the mean error ratio of EDS with classical D-S method will decrease. EDS increased from 0.245% to 0.086%. Classical D-S method decreased from 0.287 to 0.122%. But under the same number of evidences, the mean error ratio is much lower than that of the classical D-S method. The comparison result can be shown in Figure 7.

Based on the comparison curve in Figure 7, we can see that there is the different change trends and error ratio between EDS and classical D-S method for making decision by embedded UPS system. Why? Because when EDS is adopted, it has considered the reliability, time-efficiency and relativity of service context of mobile applications when control action is made by the embedded UPS system. But classical D-S method ignored these parameters. At the same time, we can see that both methods can show the correctness, and validity of control action is made by embedded UPS system but the belief degree has little difference.

With the increase of checked objects or targets, the mean error ratio dealt with by EDS, RST (Mahler 2007) and BT for making decision by embedded UPS system will decrease. EDS decreased from 0.156 to 0.048%. RST decreased from 0.201% to 0.063%. BT decreased from 0.252 to 0.119%. Our result shows their change trends and error ratio. Based on comparison, the advantage of EDS is obvious.

From the comparisons in Figure 8, we can see that the error ratio of EDS is the least, but Bayesian (probability) Theory (BT) is worst, so EDS is the most efficient among EDS, RST (Mahler 2007) and BT methods. The reason is that Bayesian (probability) Theory method only depended on basic probability assignment set by the user, so it has drawbacks about other impact factors, such as reliability, time-efficiency and relativity. Therefore, it is larger than EDS and RST. RST method also ignored the reliability, relativity of service context of mobile application. On the other hand, RST used more space and time than EDS when it did computing process. By comparisons, as we know, we can see that the three methods can show the correctness, and validity of control action is made by embedded UPS system but the belief degree has a little difference.

The relationship between Figures 6, 7 and 8 is as follows. They are different methods for testing the same experimental examples. But their target is the same, i.e.

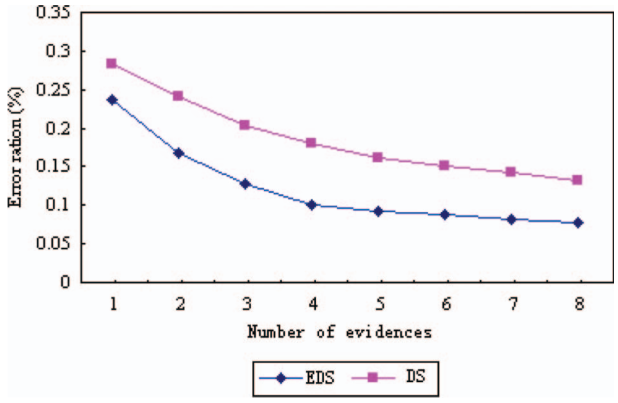

Figure 7. Comparison result of EDS and DS.

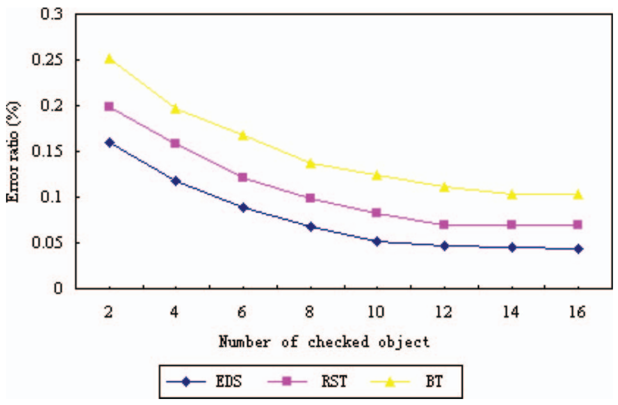

Figure 8. Comparison result of RST, EDS and BT.

to get the belief degree of control action for implemented embedded UPS system. When the belief degree is high, the control action of the system is effective. Otherwise it cannot support and meet the requirement of mobile applications. From the result of experiments and tests shown in Figure 6–8, we can conclude that the belief degree of the proposed system is high, it is valid and correct.

9. Conclusion and future works

For supporting mobile application, a design and implementation solution of EUPSS is brought forward for long-distance monitoring and controlling of UPS based on Web network. The hardware and software of the embedded UPS system is analysed and presented in this article. The system structure includes hardware structure, software structure and the accessing process from client to server. The implementation is based on ATmega161, RTL8019AS and Arm chips with TCP/IP protocol suite for communication. In the embedded UPS system, we have shown how to implement embedded file system on serial EEPROM based on I2C bus. With the limitation of the capacity of EEPROM, the number of files stored in SEROMFS is restricted. Because of serial operation, the access delay is high. In order to reduce the delay, we adopt other communication protocol, such as SPI, where the speed can reach 6Mbps. For some Web-based mobile applications, such as embedded web server, the file system is designed for storing web pages and pictures which normally don't need frequent updating, we select the Flash ROM to store files. Based on our experiments and tests, the performance of the embedded UPS meets the requirement of all kinds of Web-based mobile applications, such as B/S applications and peer to peer application in mobile process. So it is useful to the end user.

There are some limitations of the proposed system. We have implemented an embedded file system on serial EEPROM based on I2C bus. But with the limitation of capacity of EEPROM, the number of files stored in SEROMFS is restricted, so is the size of each file. Because of serial operation, the access delay is high. In order to reduce the delay, we will adopt other communication protocols, such as SPI, whose speed can be 5 Mbps.

In addition, there are a lot of works to be done to support different functions. For example, if there is a good integration development environment for EI application under EI, it will provide a friendly platform for developer. It provided high-level software development tools and low-level software for web server, support remote file download and server configuration function.

Acknowledgements

This work is supported by '863' project plan of China (No. 2007AA01Z188), National Natural Science Foundation of China (Nos. 61170173, 60773073 and 61001174), Program for New Century Excellent Talents in University of China (No. NCET-09-0895), Key project of Ministry of Education of China (No. 208010) and Tianjin Foundation (No. 10JCYBJC00500).

References

Atmel profile, 2009a. [online]. Available from: http://www.atmel.com/atmel/acrobat/doc0507.pdf [Accessed 3 October 2011].

Atmel profile, 2009b. [online]. Available from: http://www.atmel.com/atmel/acrobat/doc2396.pdf [Accessed 3 October 2011].

Atmel profile, 2009c. [online]. Available from: http://www.atmel.com/atmel/acrobat/doc0954.pdf [Accessed 3 October 2011].

Atmel profile, 2009d. *Application notes* [online]. Available from: http://www. atmel.com/atmel/acrobat/doc0951.pdf [Accessed 3 October 2011].

Atmel profile, 2009e. [online]. Available from: http://www.atmel.com/atmel/acrobat/doc0670.pdf [Accessed 3 October 2011].

Capozucca, A. and Guelfi, N., 2010. Modelling dependable collaborative time-constrained business processes. *Enterprise Information Systems*, 4 (2), 153–214.

D'Mello, D. and Ananthanarayana, V.S., 2010. Dynamic selection mechanism for quality of service aware web services. *Enterprise Information Systems*, 4 (1), 23–60.

Deborah, E. and Ramesh, G., 2008. Embedding the internet. *Communications of the ACM*, 7 (7), 38–41.

Duan, L., Street, W.N., and Xu, E., 2011. Healthcare information systems: data mining methods in the creation of a clinical recommender system. *Enterprise Information Systems*, 5 (2), 169–181.

Fisher, J., 1999. Fast JPDA multi-target tracking algorithm. *Applied Optics*, 28 (1), 371–375.

Gong, Z.G., Maybin, M., and Guo, J.Z., 2010. Business information query expansion through semantic network. *Enterprise Information Systems*, 4 (1), 1–22.

Hu, X.B., 2009. Multi-valued performance metrics for real-time embedded systems. *Design Automation for Embedded Systems*, 5 (6), 15–28.

Julio, C.T., 2008. Non-linear system modeling via online clustering and fuzzy support vector machines. *International Journal of Modeling, Identification and Control*, 4 (2), 101–111.

Kakousis, K., Paspallis, N. and Papadopoulos, G., 2010. A survey of software adaptation in mobile and ubiquitous computing. *Enterprise Information Systems*, 4 (4), 355–389.

Koch, S. and Mitlohner, J., 2010. Effort estimation for enterprise resource planning implementation projects using social choice – a comparative study. *Enterprise Information Systems*, 4 (3), 265–281.

Lavagno, L., 2007. Design of embedded systems: formal models, validation, synthesis. *Proceedings of the IEEE of Embedded Technology*, 5 (5), 173–182.

Lee, L., 2010. Internet embedded systems: poised for takeoff. *IEEE Internet Computing*, 2 (3), 24–29.

Mahavir, S., 2008. Neural network fault classification of transient data in an automotive engine air path. *International Journal of Modeling, Identification and Control*, 3 (2), 148–155.

Mahler, R., 2007. Random set as a foundation for general data fusion. *Proceedings of the Sixth Joint Service Data Fusion Symposium*, 1 (1), 357–394.

Mori, S., Chong, C.Y., and Wishner, R.P., 1998. Tracking and classifying multiple targets without a priori identification. *IEEE Transactions on Automatic Control*, 31 (5), 401–409.

Narman, P., *et al.*, 2011. Data accuracy assessment using enterprise architecture. *Enterprise Information Systems*, 5 (1), 37–58.

Paul, C. and Richard, M., 2006. Managing context data for smart spaces. *IEEE Personal Communications*, 10 (10), 44–46.

Reid, D.B., 2007. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24 (6), 843–854.

Saha, F.T. and Chang, T.C., An efficient algorithm for multi-sensor track fusion. *IEEE Transactions on Aero-Space Electronic Systems*, 34 (1), 200–210.

Satyanarayanan, M., 2001. Pervasive computing: vision and challenges. *IEEE Personal Communications*, 8 (8), 10–17.

Shen, C.W. and Chou, C.C., 2010. Business process re-engineering in the logistics industry: a study of implementation, success factors, and performance. *Enterprise Information Systems*, 4 (1), 61–78.

Wickramasinghe, V. and Gunawardena, V., 2010. Effects of people-centred factors on enterprise resource planning implementation project success: empirical evidence from Sri Lanka. *Enterprise Information Systems*, 4 (3), 311–328.

Wolf, W., 2010. Hardware-software co-design of embedded systems. *Proceedings of the IEEE Embedded Technology*, 8, 167–189.

Xu, E., Wermus, M., and Bauman, D., 2011. Development of an integrated medical supply information system. *Enterprise Information Systems*, 5 (3), 385–399.

Zhang, D.G., 2006. Web-based seamless migration for task-oriented nomadic service. *International Journal of Distance E-Learning Technology (JDET)*, 4 (3), 108–115.

Zhang, D.G., 2008. A kind of new decision fusion method based on sensor evidence. *Journal of Information and Computational Science*, 5 (1), 171–178.

Zhang, D.G., 2009. A kind of new approach of context-aware computing for ubiquitous application. *International Journal of Modeling, Identification and Control*, 8 (1), 10–17.