Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

An embedded lightweight GUI component library and ergonomics optimization method for industry process monitoring*

Da-peng TAN[1], Shu-ting CHEN[†‡2], Guan-jun BAO[1], Li-bin ZHANG[1]

[1]*College of Mechanical Engineering, Zhejiang University of Technology, Hangzhou 310032, China*

[2]*Department of Basic Medicine, Hangzhou Medical College, Hangzhou 310053, China*

[†]E-mail: shutinren@163.com

Received Oct. 24, 2016; Revision accepted Jan. 27, 2017; Crosschecked May 10, 2018

Abstract: Developing an efficient and robust lightweight graphic user interface (GUI) for industry process monitoring is always a challenging task. Current implementation methods for embedded GUI are with the matters of real-time processing and ergonomics performance. To address the issue, an embedded lightweight GUI component library design method based on quasar technology embedded (Qt/E) is proposed. First, an entity-relationship (E-R) model for the GUI library is developed to define the functional framework and data coupling relations. Second, a cross-compilation environment is constructed, and the Qt/E shared library files are tailored to satisfy the requirements of embedded target systems. Third, by using the signal-slot communication interfaces, a message mapping mechanism that does not require a call-back pointer is developed, and the context switching performance is improved. According to the multi-thread method, the parallel task processing capabilities for data collection, calculation, and display are enhanced, and the real-time performance and robustness are guaranteed. Finally, the human-computer interaction process is optimized by a scrolling page method, and the ergonomics performance is verified by the industrial psychology methods. Two numerical cases and five industrial experiments show that the proposed method can increase real-time read-write correction ratios by more than 26% and 29%, compared with Windows-CE-GUI and Android-GUI, respectively. The component library can be tailored to 900 KB and supports 12 hardware platforms. The average session switch time can be controlled within 0.6 s and six key indexes for ergonomics are verified by different industrial applications.

Key words: Embedded lightweight graphic user interface (GUI); Quasar technology embedded (Qt/E); Industry process monitoring; Multi-thread; Ergonomics performance
https://doi.org/10.1631/FITEE.1601660 **CLC number:** TP206

1 Introduction

With information technology as the dominant driving force, traditional industry has been undergoing a high-tech revolution and modern industry is developing with the inevitable trends of intellectualization, integration, and coordination (Yao et al., 2015; Liao et al., 2016; Tan et al., 2016c). By the growth and development of modern industry production, higher working performance is increasingly required by industry process monitoring systems (IPMSs). An IPMS should provide powerful functions, good real-time performance, and perfect human-computer interactivity. However, a traditional IPMS based on analogue instruments or microcontroller units (MCUs) cannot meet the requirements mentioned above.

Owing to the ability to supply sufficient support for IPMSs, the emerging embedded system (ES) has become the research hotspot of the industry monitoring area. An ES has been making IPMSs enter an intelligent period, where embedded intelligent instruments (EISs) are in the mainstream of modern

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (Nos. 51775501, 51375446, U1509212, and 51405441), the Zhejiang Provincial Natural Science Foundation, China (No. LR16E050001), and the Zhejiang Provincial Health Department Program, China (No. 2015KYA067)

ORCID: Shu-ting CHEN, http://orcid.org/0000-0002-4101-0649

IPMSs. Apparently, as an indispensable section of EIS, the lightweight graphic user interface (GUI) adapted to ES environments should receive much attention (Wang et al., 2015; Wu et al., 2016).

An ES is a special computer system that is embedded into the host machine and focuses on special applications, including the industry monitoring area, as shown in Fig. 1. Therefore, the embedded software and hardware should be customized, tailored, and configured dynamically based on different application requirements, and should meet strict technical requirements in the areas of function, reliability, cost, volume, and power consumption (Zheng et al., 2015; Tan et al., 2016b; Chen and Tan, 2018). Compared with a traditional computer system, an ES has the following features: high real-time performance, low power consumption, wide versatility, and adequate interactivity. Moreover, it can provide stable, quick, and reliable services under special conditions. Therefore, an ES has been used in many industrial areas, such as traffic control (Barrero et al., 2010), intelligent household appliances (Steblovnik and Zazula, 2011), intelligent building management systems (Saponara et al., 2011), point of sale (POS) networks (Ramos and Penteado, 2008), biological engineering (Mazzei et al., 2008), advanced manufacturing (Tan et al., 2013a, 2016c; Ji et al., 2017; Li et al., 2017), robot control, and manufacturing process monitoring (Tan et al., 2010; Li et al., 2014). Many researchers have studied the industry monitoring systems, such as data-driven monitoring (Yin et al., 2015) or sensorless detection (Tan and Zhang, 2014; Tan et al., 2017b), in which an EIS is an indispensable physical carrier. An EIS generally consists of an embedded micro-processor, peripheral devices, multi-channel sensors, an embedded real-time operating system, and related application software, where the lightweight GUI belongs to the embedded application software category (Jin and Wu, 2008). Although the GUI is not in the kernel section of an EIS, it is the last executing segment for users, and it directly influences the interaction efficiency and working performance of an EIS.

As indicated above, GUI is an indispensable part of an embedded IPMS. Because an ES has strict hardware/software demands, it addresses higher functional requirements for GUI (the so-called embedded lightweight GUI), which can not only adapt to embedded environments but also provide customized function interfaces for an IPMS. Therefore, an embedded lightweight GUI should occupy little storage space, be customized and tailored, and provide high reliability and robust performance (Drossu et al., 1996; Zeng et al., 2013). With respect to the requirements above, there are many types of graphic tool libraries that can perform the development of an embedded GUI, such as Micro-windows, GTK, Xfree86, MiniGUI, Windows-CE-GUI, and quasar technology embedded (Qt/E).

1. Micro-windows. It is generally adopted on a small display unit and can be compatible with different embedded platforms (Zhou and Xiang, 2013). Moreover, it is based on the client/server (C/S) pattern, with a simple source code and a rapid switching speed, requiring little system resource. However, its network transparency needs to be improved and it has fewer application program instances and introductions than the other graphic tool libraries considered.

2. GTK/GTK+. It is with platform-independent performance including microcontrollers and needs less storage resource. GTK/GTK+ is common for X-Windows, so it needs to access the X-server, which might limit the working efficiency (Ahn et al., 2006).

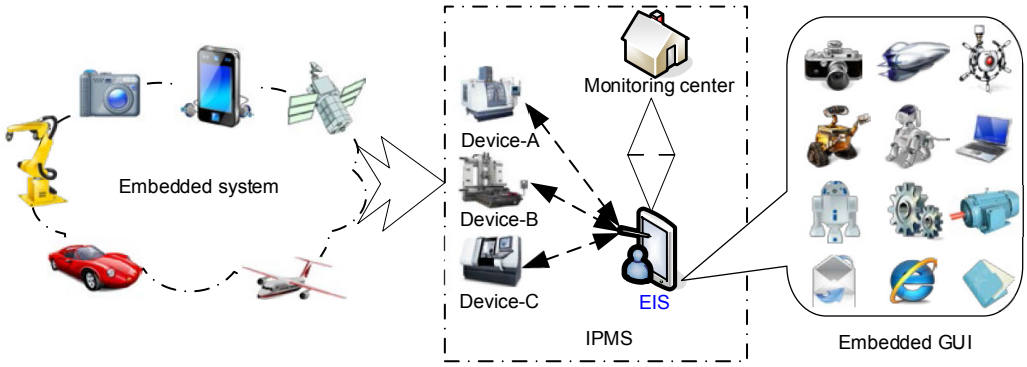

Fig. 1 Embedded system and lightweight GUI of a modern IPMS

3. Xfree86. This is an open-source X-Windows system that can be distributed freely and repetitively, support the frame buffer architecture, and possess a lot of online technical documents and easily extendable application program interfaces (APIs). Compared with Micro-windows, however, its normal running process requires more internal memory space resource.

4. MiniGUI. Developed by Y. M. Wei, MiniGUI can support embedded real-time systems. It runs mainly on a Linux operating system, but it is also compatible with the portable operating system interface of unix (POSIX) standard framework. It requires small memory space, and is easy to configure and transplant according to different embedded platforms (Zhang et al., 2013).

5. Windows-CE-GUI. It is an embedded edition of Microsoft Windows, with an open 32-bit architecture. It is actually a simple Windows 95 and is specially designed for personal digital assistants (PDAs) and other mobile electronic terminals. Windows-CE uses the hardware abstraction layer (HAL) to configure hardware resource. Therefore, it can obtain perfect technical performance, including the lightweight GUI. Windows-CE-GUI can provide powerful development tools, but it is a protected commercial software project (Rehault, 2010).

6. Qt/E. Qt is a mature GUI tool library, has perfect cross-platform ability, and can be applied in presently popular operating systems (OSs), such as Windows, Linux, Solaris, FreeBSD, QNX, HP-UX, and Irix. Based on different application environments, Qt can be divided into two categories: the X11 version (Qt/X11) for X-Windows and the embedded version (Qt/E) for ES environments (Riskedal, 2008). Qt/E is based on the original Qt and takes adequate function adjustments to adapt to ES environments. Its hierarchical structure is shown in Fig. 2.

Qt/E can communicate directly with I/O devices. Moreover, it has an object-oriented design (OOD) architecture that makes its code well-organized, reusable, and work with a fast switching speed. As an embedded lightweight GUI development tool, Qt/E accommodates the frame buffer driver. Therefore, it can write directly to the frame buffer under the condition of no X-server or X-library. In this way, it saves memory space and improves the running efficiency of the GUI effectively (Dalheimer and Hansen, 2002). In view of these technical advantages of Qt/E, we select it as a fundamental tool to develop an embedded lightweight GUI for an IPMS.

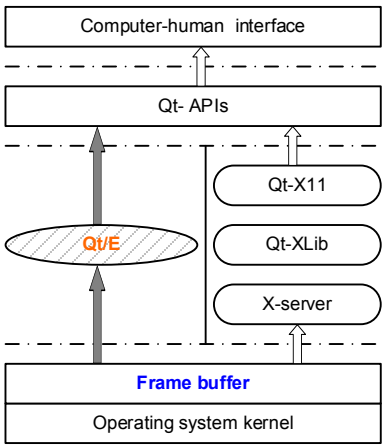

Fig. 2 System hierarchical structure of Qt/E

There are several types of beautiful and friendly GUIs such as embedded Wizard, μGFX, android-GUI, and IOS-GUI. They have been applied to mobile phones or panel computers, but are rarely used in embedded instruments or industrial controllers. The reason is that GUI could influence their normal workings. Therefore, an embedded GUI development for an IPMS is always a challenge.

As indicated above, the embedded lightweight GUI is of great significance for IPMSs, and there are dozens of development tools with different features. However, according to the descriptions in references (Drossu et al., 1996; Riskedal, 2008; Rehault, 2010; Ji et al., 2010, 2012; Zeng et al., 2013; Zhang et al., 2013; Zhou and Xiang, 2013; Cecotti, 2016; Tan et al., 2017a), there are some problems for practical industry applications, such as occupying a large space, low stability, poor configurability and cross-platform capability, and poor real-time performance. Moreover, the above existing methods for embedded IPMS-GUIs are with the matters of session switching and ergonomics performance. Therefore, to address the matters, we propose a novel Qt/E-based embedded lightweight IPMS-GUI component library design method and an ergonomics optimization method.

To reach the above research target, the main scientific/technical contributions of this study are as follows: (1) An entity-relationship (E-R) model oriented to the IPMS-GUI component library is

developed to define the functional modules' framework and data coupling relations; (2) To provide support for different embedded computer architectures, a cross-compilation environment platform that can adapt to multiple embedded targets is constructed, and the Qt/E library files are tailored to satisfy the running requirements of the embedded target environments; (3) Using the Qt/E-based signal-slot communication interfaces, a message mapping mechanism that will not require a call-back pointer is realized, and the context switching speed is improved; (4) Using a multi-thread method, a status transmission model for distributed real-time tasks is established, the parallel digital signal processing abilities of the IPMS-GUI are enhanced, and its real-time performance and robustness are guaranteed; (5) Based on the modularization principle, combined with the self-designed monitoring signal processing algorithms, a universal embedded IPMS-GUI shared component library is developed, which can improve the cross-platform applicability of the lightweight GUI, and is capable of being configured and tailored for different monitoring objects; (6) The human system interactive process is optimized by a scrolling page method, and the ergonomics performance is improved.

2 Embedded lightweight GUI framework and cross-compilation environment

2.1 Embedded lightweight GUI framework

It is well known that OOD and modularization are kernel facets of modern software development. Therefore, the key issue of lightweight GUI component library design is how to construct a framework.

A component is regarded as a special software entity that is of specific structure and that functions in accordance with related industrial standards, can perform different services in response to different user requests, and can be independently assigned and encapsulated. The software production pattern that adopts the component technology is similar to the mechanical workpieces assembly process, which introduces the standard workpieces, production streamlining, and systematic assembly definitions into software area, and breaks the manual workshop software development pattern. By the component-based design method, software is apt to be reused, refined, optimized, and has good configurability (Wulf et al., 2008; Xu et al., 2012; Zeng et al., 2016).

As mentioned above, a component is a software entity, and the data coupling of different components can be regarded as the relationship. Therefore, the E-R model can effectively express the topological structure of a component cluster. The corresponding structural optimization method will then be used to improve the working performance of the component library. Subsequently, according to the fundamental requirements of an IPMS, combined with the component principles, an E-R model oriented to the embedded IPMS-GUI component library is set up as in Fig. 3. Considering the task requirements of an IPMS, besides the functions of graphic display and control information interaction, the GUI component library requires the abilities of real-time data calculation and management to finish the monitoring tasks, which are a kernel characteristic different from other general-purpose GUIs. Therefore, the component library can be divided into the following software modules: data management module (DMM), analysis calculation module (ACM), graphic show module (GSM), and system control module (SCM).

DMM receives real-time data packets that encapsulate a series of physical signals from target devices and are uploaded by the data acquisition module (DAQM). Then DMM unpacks the data packets into different data elements according to the self-defined data transfer protocol, and distributes the data elements to ACM. DMM can provide the interface history data playback needed to perform the correlation analysis between real-time data and history data, which is an effective measure of the working status of the target device. Moreover, DMM can supply the file management interfaces, such as storage, query, addition, combination, comparison, and deletion.

After receiving the real-time monitoring data, ACM executes the digital signal preprocessing tasks, including software filtering, fast Fourier transform (FFT), and power spectrum analysis, and then transfers the preprocessed signal to GSM. GSM possesses the core modules of the GUI component library and provides the following six types of graphic interfaces: time-domain figure (TDF), frequency domain figure (FDF), stick shape figure (SSF), axes track figure (ATF), battery power figure (BPF), and static parameter figure (SPF). GSM depends on the calculation

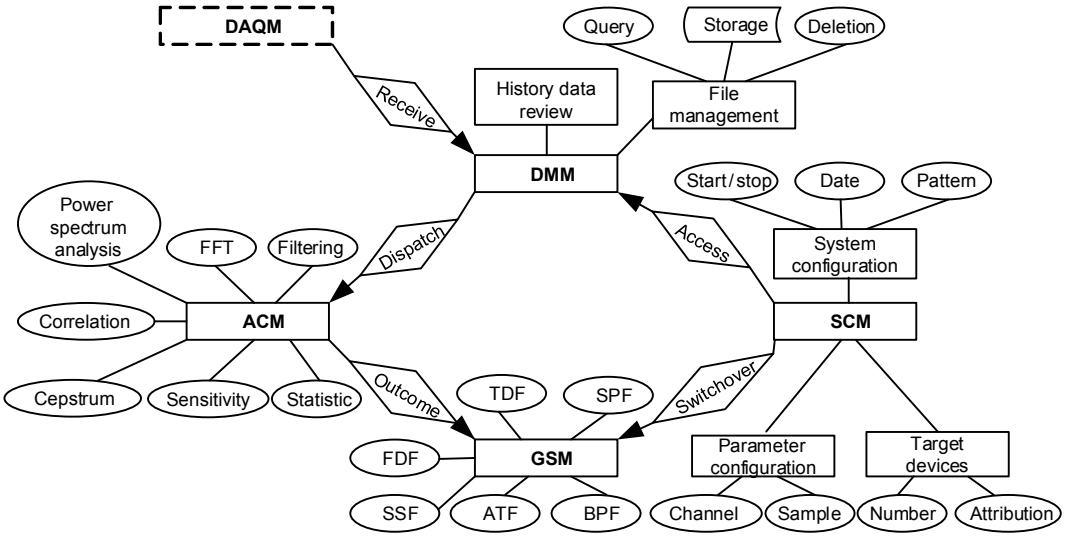

Fig. 3 Entity composition and data coupling of the embedded GUI component library

results of ACM and receives control instructions from SCM to finish the interface switching. SCM provides a system control dashboard, monitoring demand confirmation, target device configuration, and other function interfaces for different user sessions.

Based on the above GUI framework, using the OOD method, all the software modules are encapsulated by universal parameter interfaces. Therefore, the data coupling between different modules is reduced, the cohesion performance is increased, and the development efficiency of the GUI component library will be guaranteed. Accordingly, users can customize the function interfaces based on different industry monitoring requirements and do not need to know the internal structure or processing procedures. In this way, the flexible graphics services for industry monitoring are realized, and the applicable performance of the IPMS will be improved.

2.2 Cross-compilation development environment

Because this study is oriented to the ES environment, the provision of an embedded cross-compilation development environment for the Qt/E-based GUI library is an indispensable task. Therefore, after defining the E-R framework of GUI, the following operation shows how to construct a Qt/E cross-compilation development environment that can adapt to different embedded architecture platforms. As Fig. 4 shows, the cross-compilation environment is a code generating tool-chain in the host computer (local PC system). It can construct the hardware/software conditions of embedded target systems, build a local compiler based on the source code and embedded library files, and generate the corresponding application software and user drivers. Subsequently, the software and drivers are downloaded into the embedded target system to execute various industry monitoring tasks.

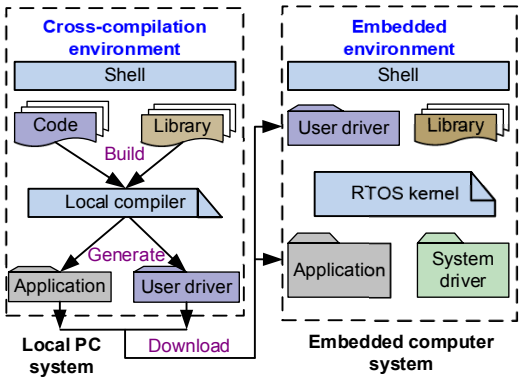

Fig. 4 Embedded cross-compilation development environment (RTOS: real-time operating system)

As mentioned above, the executable files of an embedded program developed by the cross-compilation environment cannot run on the host computer directly; therefore, it should be transmitted to the target embedded system to check that the resulting system runs correctly. If mistakes occur, one must revise the source code, and compile and download the executable files again, until the destination is

satisfied. Considering the above method, the development and tests are two absolute processes; thus, it is inconvenient to develop and maintain the embedded application software, especially for the user test procedures.

To solve this problem, we propose a virtual embedded target platform in the local PC system that can simulate the working conditions of the target system and accomplish online programming and debugging. In this circumstance, only after all expected functions satisfy the design requirements, are the executable files permitted to be downloaded into the target system. This method can effectively improve the development efficiency of the embedded software, and the establishment procedures are described in Fig. 5.

2.2.1 Local virtual environment

It is complex to simulate the hardware/software conditions of the embedded target on the local PC system. In allusion to the problem, Qt/E offers a convenient simulation tool, the console of the Qt virtual frame buffer (QVFB) (Figs. 5 and 6). It can simulate most of the popular ES frameworks by configurations of geometry, size, and color depth.

First, a Qt/E image file is extracted to the Qt root directory ($QTDIR), the environment variables should be set, and the -qvfb option should be appended when the configuration is executed to generate a complication guide file (Makefile). Then after compiling the Qt/E image file, a QVFB executable file can be found in the related directory. If an application program has been implemented, the running results can be verified in the QVFB console with the Qt/E Windows server (QWS) (Li et al., 2015; Lin et al., 2015; Tan et al., 2016a). Therefore, QVFB can perform on-line programming and debugging for embedded application software, and improve the embedded software development efficiency. Moreover, it can solve the competitive conflicts for software/hardware resources and realize parallel development for dependent modules of the embedded GUI component library.

2.2.2 Embedded cross-compilation environment

An embedded cross-compilation environment is decided by different embedded platforms. This study takes the PXA-988 advanced RISC machine (ARM) processor and real-time Linux (RT-Linux) as an instance to indicate the establishment procedures.

First, the ARM-Linux cross-compilation tool chain, including arm-linux-binuitls, arm-linux-glibc, arm-linux-gcc, and arm-linux-knl, is downloaded and installed into the designated directory, and the environment variables need to be appended and revised.

Second, according to the PXA-988 platform characteristics, Qt/E is configured by the command with the following parameters: /configure -xplatform arm-linux-gcc -shared -noxft -noqvfb -libpng -libjpeg -zlib -vnc. These parameters stand for the configuration attributes for the embedded target and provide technical support as follows: multi-platform, ARM architecture, shared library, and graphic figure.

Finally, after system configuration, the Makefile is generated to compile Qt/E and create corresponding embedded shared library files and other executable programs.

Based on the above method, a simulated PDA software is generated (Fig. 6) that can simulate the working conditions of PDA and provide integral support for Qt/E.

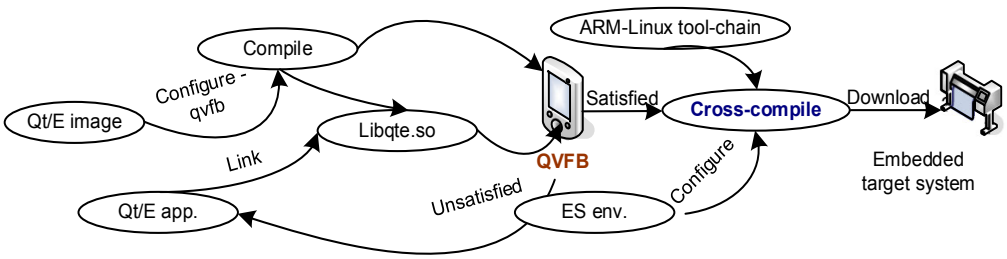

Fig. 5 Cross-compilation environment establishment flow for Qt/E-based IPMS-GUI

Fig. 6 Virtual embedded platform for lightweight GUI development: geometry size 240 px×320 px and color depth 16

3 Qt/E component library tailoring and monitoring processing algorithms

3.1 Qt/E library tailoring

Owing to the strict requirements of the ES environment, the Qt/E library files need to be tailored to reduce their storage and running space. Therefore, the tailoring of the Qt/E library is a critical step for the development of the GUI component library. It determines whether the component library can meet the fundamental requirements of an embedded IPMS and will have significant influence on subsequent monitoring processing algorithms' loading.

Qt/E provides abundant widgets and APIs, of which the embedded GUI requires only a few. With the design tasks for the GUI component library done, the classes that may be called should be confirmed according to the E-R framework shown in Fig. 3, and the other classes will be tailored to save memory space and improve the program's working efficiency. The functional requirements of IPMS-GUI are of a relatively low level that needs only the basic framework of Qt/E: common widgets, graphic frame, drawing components, event mechanism, network protocol, multi-thread, and database. Therefore, enough memory space for the digital signal processing algorithms can be saved. The structure of the classes of the tailored Qt/E is shown in Fig. 7.

Considering the tailoring process, the widgets that will be deleted should be confirmed first and defined by the shutdown macro. The header file

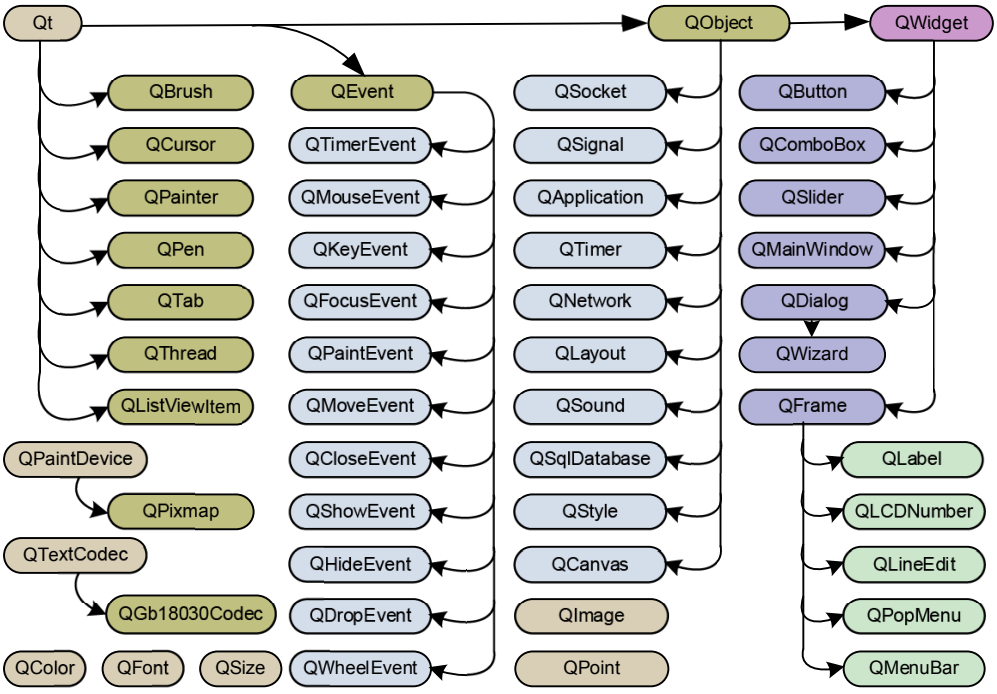

Fig. 7 Class structure of the tailored Qt/E for an IPMS

'$(QTDIR)/src/tools/qfeatures.h' contains all macro definitions and their affiliated relationship. Because of the OOD ability of Qt/E, if a macro is selected, all of its child macros will be activated. Therefore, the affiliated relationships of different macros need to be confirmed. The header file '$(QTDIR)/include/qconfig.h' is the tailoring configuration file, where it is defined whether the related widgets are to join in the compilation.

Since the macro format is with the scheme of QT_NO_DIR, if it is defined and opened in 'qconfig.h', the corresponding widget will be tailored, i.e., not permitting the widget to be compiled. The default Qt/E library is the maximum compilation image file (7 MB), and it can be reduced to 640 KB at least. When compiling the Qt/E tailored, one must override the original 'qconfig.h' with a new, modified one (Dalheimer and Hansen, 2002; Lin et al., 2015). We choose Qt/E-2.3.7 as the tailoring object, whose shared library files (including network, multi-thread, and fonts) occupy only 900 KB of the internal memory space. Qt/E-2.3.7 can run well on the ARM platforms of the PXA-988 (533 MB memory) and the S3C6410 (64 MB memory), and it match the demands of the current ES platforms.

3.2 Library file loading of Qt/E classes and monitoring algorithms

3.2.1 Qt/E library file loading

When the Qt/E compilation is finished, a shared library file 'libqte.so.2.3.7' will be created. In addition, there are three soft links: 'libqte.so.2.3', 'libqte.so.2', and 'libqte.so'. The three links are very important. They have the same source file 'libqte.so.2.3.7'. Considering the keywords conflict of different Qt/E versions, they are the indispensable searching objects during the course of program linkage and execution. When an embedded application program is executed, it will call the shared library files that should be downloaded into the embedded target system and generate the above three links.

To prevent conflicts between the user library files and system library files, it is necessary to create a new directory to store the user library files. In reference to the above operations, a shared library guide file 'ld.so.conf' should be appended, containing the library file guide information for the execution of the application programs. Subsequently, the 'ldconfig' command should be executed, which can update the contents of 'ld.so.conf' and make the new user library files effective. Similarly, the library files of the monitoring algorithms can be loaded in the same way (Fig. 8), which generates the shared library 'libdig.so' without soft links that are described in the next subsection.

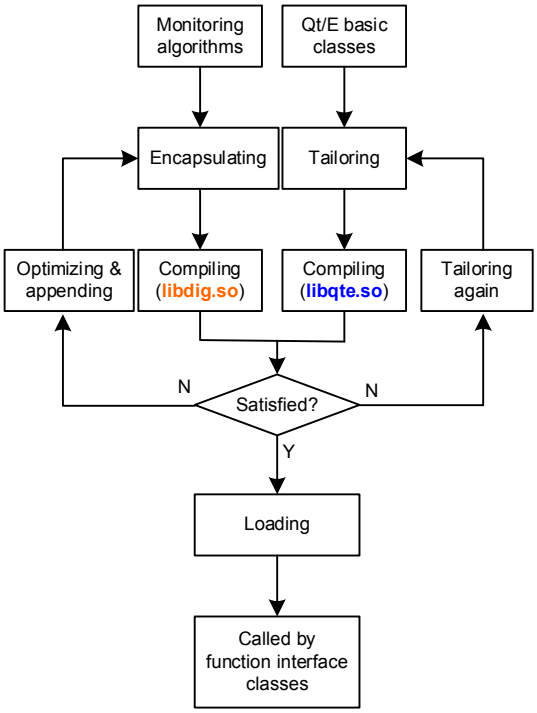

Fig. 8 Library files loading of Qt/E classes and monitoring algorithms

3.2.2 Algorithm encapsulation and loading

Considering the fundamental requirements of an IPMS, the monitoring algorithms are indispensable, taking responsibility for status monitoring and default diagnosis. The monitoring algorithms provide a series of data resources for different graphic interfaces; therefore, they are the kernel background programs of an IPMS.

The IPMS-GUI component library includes the following monitoring algorithms: (1) six digital signal pre-processing methods including infinite impulse response (IIR) filter, finite impulse response (FIR) filter, average value filter, removal zero drift, Hamming window, and interpolating coding; (2) six time-domain analysis methods including time sequence analysis, status transmission analysis, self correlation,

interactive correlation, axes track, and sliding motion calculation; (3) seven frequency domain methods including FFT, short-time Fourier transform (STFT), power spectrum density (PSD), discrete wavelet transform (DWT), wavelet packet (WP), Hilbert-Huang transform (HHT) (Veltcheva and Soares, 2012), and three-dimensional frequency spectrum analysis.

Moreover, the embedded GUI library can provide 12 single-value signature algorithms, which characterize the statistic symbols of the real-time signal sequence and have the advantages of deep impression, high effectiveness, and strong real-time performance (Tan et al., 2013b). The 12 single-value signatures contain maximum (S_1), minimum (S_2), peak-to-peak (S_3), mathematical expectation (S_4), root mean square (S_5), standard deviation (S_6), skewness (S_7), kurtosis (S_8), frequency center (S_9), mean square frequency (S_{10}), frequency variance (S_{11}), and signal intensity (S_{12}).

S_1 is the maximum value of the discrete sampling sequence and contains a local maximum and a global maximum (Eq. (1)). The first one is for the present sampling sequence and the other considers the whole sampling time segment. S_2 represents the minimum of the sampling sequence, consisting of a local value and a global value. S_3 is the difference of S_1 and S_2; it characterizes the absolute value variation of the sampling sequence. S_4 is the average value of the sampling sequence, and can act as a basic variable for calculating other signatures.

$$S_1=\begin{cases}\max(x_i,x_{i+1},...,x_j), & 0\le i<j\le N,\\ \max(x_i), & 0\le i\le N,\end{cases} \tag{1}$$

$$S_2=\begin{cases}\min(x_i,x_{i+1},...,x_j), & 0\le i<j\le N,\\ \min(x_i), & 0\le i\le N,\end{cases} \tag{2}$$

$$S_3=S_2-S_1, \tag{3}$$

$$S_4=\overline{x}=\frac{1}{N}\sum_{i=0}^{N-1}x_i, \tag{4}$$

where N is the length of the sampling sequence, x_i is the amplitude of one discrete sampling point, and $\overline{x}$ is the average of the sequence.

S_5 represents the root mean square (RMS), and can reflect the intensity and effective value of the physical signal. S_6 is the standard deviation, and can characterize the dispersion degree of the discrete sampling sequence. S_7 represents a skewness coefficient, which is a profile parameter and describes the dissymmetry degree of the probability distribution.

$$S_5=\sqrt{\frac{1}{N-1}\sum_{i=0}^{N-1}x_i^2}, \tag{5}$$

$$S_6=\sqrt{\frac{1}{N-1}\sum_{i=0}^{N-1}(x_i-\overline{x})^2}, \tag{6}$$

$$\begin{aligned}S_7&=\frac{N\sum_{i=0}^{N-1}(x_i-\overline{x})^3}{(N-1)(N-2)S_6^3}\\&=\frac{N}{(N-1)(N-2)}\cdot\frac{\sum_{i=0}^{N-1}(x_i-\overline{x})^3}{\left[\frac{1}{N-1}\sum_{i=0}^{N-1}(x_i-\overline{x})^2\right]^{\frac{3}{2}}}.\end{aligned} \tag{7}$$

S_8 is a dimensionless signature that describes the normalized fourth-order central distance of the discrete sequence. It can express the distortion of probability density, and is apt to be influenced by the instant shock ingredients. Therefore, it can be used to filter out the shock signal with distortion characteristics. S_9–S_{11} are the signatures in the frequency domain, representing the centroid, mean square, and variance, respectively. They can reflect the profile variations of power spectrum of the discrete sequence. Regarding the real-time signal sequence of the target device, if any frequency ingredient change is caused by external factors, the frequency geometric center will correspondingly shift from the original position. Subsequently, if the number of total frequency components is increasing, the density profile of power spectrum will take on a dispersing phenomenon. Accordingly, S_9, S_{10}, and S_{11} can express the structure profile variation of the frequency domain of the sampling sequence. Assuming the sampling frequency is f_s, ρ_1 and ρ_2 are the coefficients of autocorrelation.

$$\begin{aligned}S_8&=N\sum_{i=0}^{N-1}\left(\frac{x_i-\overline{x}}{S_2}\right)^4\\&=\frac{1}{N}\sum_{i=0}^{N-1}(x_i-\overline{x})^4\left[\sum_{i=0}^{N-1}(x_i-\overline{x})^2\right]^{-2},\end{aligned} \tag{8}$$

$$S_9=\frac{\pi/2-4\rho_1/\pi}{2\pi f_s}, \tag{9}$$

$$S_{10}=\frac{\pi/3-4\rho_1+\rho_2}{4\pi^2 f_s^2}, \tag{10}$$

$$S_{11}=S_{10}-S_9^2=\frac{\pi^2/12-16\rho_1^2/\pi^2+\rho_2}{4\pi^2 f_s^2}, \tag{11}$$

where

$$\rho_1=\sum_{i=1}^{N-1}x_i\cdot x_{i-1}\left(\sum_{i=0}^{N-1}x_i^2\right)^{-1}, \tag{12}$$

$$\rho_2=\sum_{i=2}^{N-1}x_i\cdot x_{i-2}\left(\sum_{i=0}^{N-1}x_i^2\right)^{-1}. \tag{13}$$

S_{12} is the intensity coefficient, which is not apt to be controlled by frequency and directly characterizes the energy strength of the physical signal. It has an inherent relationship with the signal power:

$$S_{12}=\sqrt{\frac{1}{N}\sum_{i=0}^{N-1}\left[V(i)-\bar{V}\right]^2}. \tag{14}$$

In Eq. (14), $V(i)$ is the velocity function of the sampling sequence and $\bar{V}$ is the average value of the equivalent velocity. For example, if the sensor of an IPMS is a vibration acceleration sensor, the physical signal collected will be a continuous sequence with piezoelectric pulses. The physical signal is transformed to a digital velocity signal (to be mentioned in Section 4).

$$\because V(i)=\sum_{j=0}^{i}A(j)\cdot\frac{g}{\Phi f_s}, \tag{15}$$

$$\therefore \bar{V}=\frac{1}{N}\sum_{i=0}^{N-1}\left[\sum_{j=0}^{i}A(j)\cdot\frac{g}{\Phi f_s}\right], \tag{16}$$

$$\begin{aligned}\therefore S_{12}&=\sqrt{\frac{1}{N}\sum_{i=0}^{N-1}\left[V(i)-\bar{V}\right]^2}\\&=\frac{g}{\Phi}\sqrt{\frac{1}{N}\sum_{i=0}^{N-1}\left\{\sum_{j=0}^{i}\frac{A(j)}{f_s}-\frac{1}{N}\sum_{i=0}^{N-1}\left[\sum_{j=0}^{i}\frac{A(j)}{f_s}\right]\right\}^2},\end{aligned} \tag{17}$$

where g is the gravitational acceleration, Φ is a quantization coefficient (Φ=204.7 m/s^2), and $A(j)$ is the discrete acceleration sequence processed by an A/D converter. With respect to the four groups of monitoring algorithms, the monitoring algorithms can be defined into four classes, in which every algorithm is encapsulated with a particular manual introduction and unified parameter interface. The algorithms are programmed by only fundamental C/C++, saved as pure text files. Therefore, after compiling the source files, the shared library file 'libdig.so' requires less than 200 KB of memory space, and can adapt to the currently popular ES platform environments well.

4 Multi-thread task scheduling and signal-slot communication mechanism

4.1 Multi-thread task scheduling method

As mentioned above, real-time data acquisition, A/D transformation, digital signal processing, and graphic display are indispensable functions for an embedded IPMS; therefore, each module interface is required to have the switching ability without a long interrupt time. Moreover, the correlation analysis, comparative analysis, history data playback, and real-time data storage need to be synchronously performed. So, an IPMS should have parallel transaction processing capability to satisfy the practical requirements of industry production.

Concerning the factual operation of process monitoring, parallel task scheduling is commonly implemented by the approach of multiple parallel. In the hypothesis of the UNIX framework, if a process is to perform a task that might occupy the same data resources of itself, it will generate a child process by the so-called fork method.

Generally, fork makes an integral process image copied from the parent process to the child process. In the circumstance with a large number of tasks, the system's overhead cost tends to be higher. This is not suitable for embedded programs. When a child process is forked, the inter process communication (IPC) method is commonly adopted to transmit the internal data between the parent and child processes (Li et al., 2012). The transmitting operation from parent to child is easy to realize. While the reverse operation is hard to perform, it is easy to create zombie processes that have a possibility to cause system resource loss.

To solve the above problems, a multi-thread method is put forward to separate the source-owning property and utilization authority, and to perform parallel task processing under the condition of process resource conflicts. Therefore, thread will have better processing efficiency than the fork (Zhuo et al., 2002; Zhang and Kang, 2011; Su et al., 2014). For this study,

multi-thread operations can be implemented by the QThread class.

With respect to the basic requirements of an IPMS, a complete GUI library should offer the following functional interfaces: system management, task configuration, TDF, FDF, SPF, histogram, system power management, and file management. During file management, real-time data acquiring/processing, graphic drawing, and data storing require multi-thread operations. If a main thread is started, the sampling parameters and monitoring symbols are configured for a specific target device, and the graphic drawing thread is evoked by internal variables (Fig. 9).

4.2 Signal-slot communication mechanism

The signal-slot internal communication method is the key feature of Qt/E. For the recall operation of message mapping, it adopts integral encapsulation to substitute complex function pointers and builds up a transparent communication mechanism. That is to say, users do not need to know the internal variables and the message mapping between visual objects could be performed automatically. As the basic member functions of all Qt/E classes, signals and slots are connected by a static function connect().

Accordingly, a message mapping mechanism for industry monitoring is proposed using the signal-slot communication interfaces. It does not require a call-back pointer, and the real-time processing and data storage tasks can be called by an awakened drawing thread. By this way, if required to switch, an executing thread task can exit freely, and then a new thread will be started. The above method can make the objects own their special message transition channel, so the processing efficiency for multiple task operations will be improved.

Taking the time-domain task as an instance, as illustrated in Fig. 10, the state transition procedures can be described as follows: (1) If the time-domain task is selected, it enters the initial state and performs operation to set values for member variables and to obtain the required memory space, referring to current sampling parameters. (2) If the task receives the ready signal, it enters the data processing state to generate the discrete data sequence and display the data curves. If the task receives the unselected signal (e.g., the frequency task is evoked), the current task will exit and start the frequency task. If it receives the ready signal continuously, it will continue performing the time-domain task until receiving another signal. (3) When the task is finished, it receives the unselected signal, and then goes back to the start state. (4) If the current data sequence needs to be stored, the data storage task will be executed. The data sequence will be stored, and the storage task will wait for the next signal.

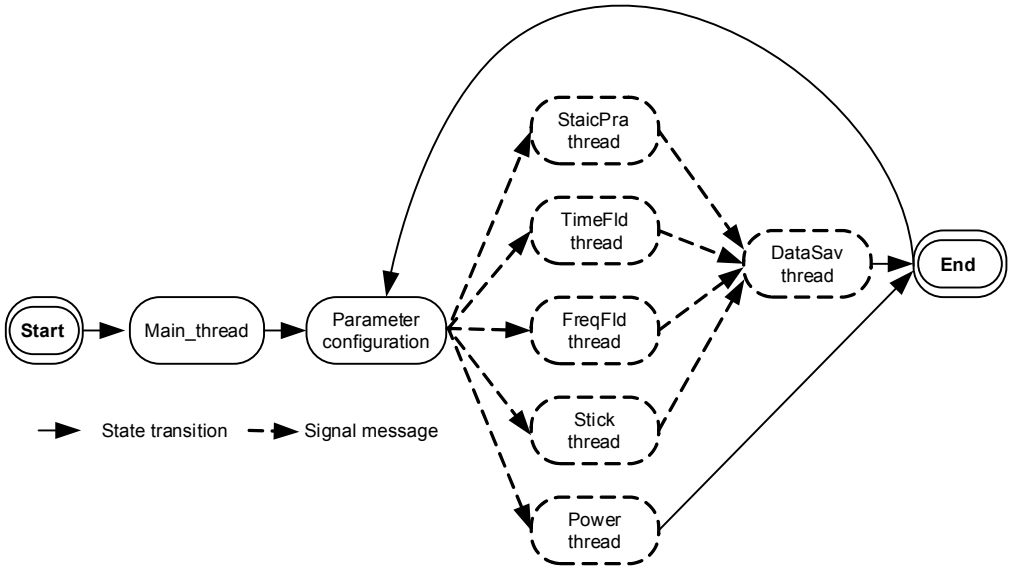

Fig. 9 Sketch map of the multi-thread task scheduling method

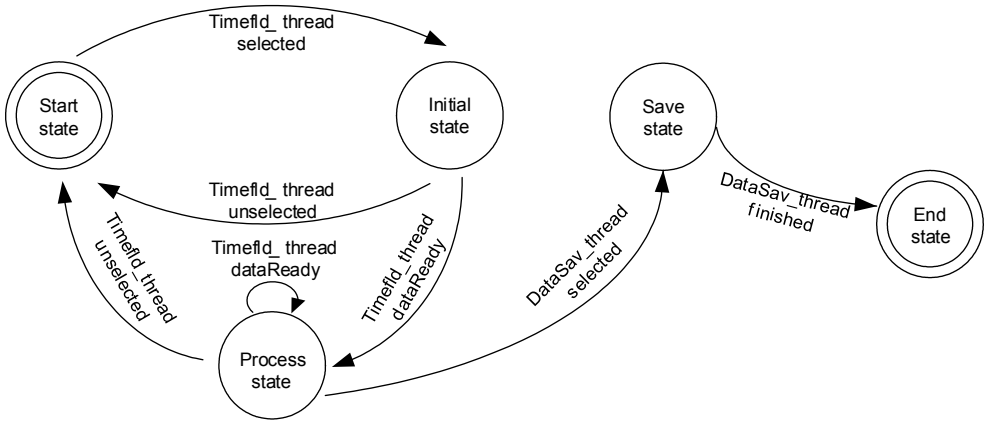

Fig. 10 Sate transform of the time-domain task by the signal-slot mapping mechanism

Since the proposed signal-slot communication method does not require a call-back pointer, it can construct the complete OOD encapsulation for a message signal and improve the security and intelligent performance of signal transmission. Therefore, the context switching speed and stability of the embedded lightweight GUI are increased and the human-computer interaction efficiency is improved.

Based on the above scheme, the mechanism can send signals directly to the 'this' pointer of the current mainframe, which can dynamically trace the instance variation and realize a seamless combination of OOD and process-oriented design (POD), such as the hybrid coding for C and C++. Moreover, it can provide the multiple signal mechanism priority for any object; therefore, users can define their own signals and slots according to specific monitoring requirements.

5 Ergonomics optimization method and multi-language environment

5.1 Human-computer interaction performance optimization method

Ergonomics performance is one of the most important technical indexes of an embedded lightweight IPMS-GUI. It determines the working effectiveness and friendliness grade and considers mainly the following factors: visual object placement, interface operating sequence, and data coupling relationship between different columns (Chevalier and Kicka, 2006; Acciani et al., 2011).

Related to the habitual operations of users, visual object placement is a key matter to influence the ergonomics performance of embedded GUIs. Accordingly, we take visual object placement as an instance to illustrate the optimization process of ergonomics performance. First, the embedded IPMS main interface that adopts the traditional menu-style object placement scheme is developed by the GUI library (Fig. 11a), where all program interfaces are working with a menu pattern and are placed in the proper order according to their importance. If a user wants to set the sampling frequency, she/he should select the parameter configuration menu (Para.) and enter the menu, where there are five settable items: monitoring device number (Dev No.), sampling parameters (Smp_P), working pattern (Pattern), sampling precision (Precision), and sampling date (Date). Then she/he should select the item Smp_P, and the sub-level menu will pop up, where there are five settable items: sampling frequency (Freq.), sampling points (Point), original phase (Phase), physical signal channel number (Chan.), and sampling begin/end time (Time). Finally, she/he should select the item Freq. to finish the operation of setting the sampling frequency. Obviously, the menu scheme possesses a clear hierarchical structure, which is the commonest software operating method in PC environments. However, it requires the operator to be familiar with every interface location. This cannot provide adequate operating efficiency, especially under the limited resource conditions of ES environments.

To address this matter, according to the investigation results of industrial monitoring demands and user operating habits, we propose a scrolling page method to optimize visual object placement and improve the operating performance of IPMS-GUI. As

shown in Fig. 11b, all program interfaces are divided into several categories. Every category performs as a scrolling page and can be switched by the red triangle button at the right-upper corner. The scrolling pages are realized by the class QTabwidget. The scrolling page structure contains more transversal space and can reduce the longitudinal depth of the menu's hierarchical structure, to improve the operating performance. In the figure, if a user selects the Para. page, a push-pull style functional interface layer tree will be shown, which can be switched by the two bilateral red triangle buttons. The user then selects the corresponding node (Freq.) of the tree to complete the task operation planned, where the interface tree is realized by the class QListview. Moreover, in the affecting frames of QTabwidget and QListview, GUI focus can respond to the coordinate variation of the touch screen, and the querying efficiency for application interfaces will be improved.

5.2 Chinese information show

It is well known that displaying clear and regular fonts with pictographic characters, such as original complex Chinese, simplified Chinese, Japanese, or Korean, is difficult on the embedded platforms. Considering Chinese users, showing Chinese information is an indispensable function of the embedded lightweight IPMS-GUI. It is well known that Chinese is an old picture-writing language that contains more than 27 000 words in the Chinese GB-18030-2000 character set and dozens of fonts. In this study, Chinese is expressed by a series of independent dot arrays. A complete Chinese library including words and fonts can occupy 10 MB or more of memory space. Therefore, it is not adaptable to ES environments.

To solve this problem, we develop a compact Chinese library. An IPMS and its log-file require only a few words that can be changed for different monitoring objects. Therefore, the compact Chinese library can be divided into 10 specialty segments: mechanic, electronic, information, computer, transport, chemistry, medicine, metallurgy, optic, and construction. The library provides only one type of Chinese font. The average number of words of the 10 segments is less than 300, i.e., only 1/10 of a mobile phone's word library. If the monitoring object is confirmed, the corresponding segment will be selected to undergo the compilation process based on the pre-compile variables.

Qt/E supports Chinese words and can provide Unicode characters with 16 bits. Therefore, Chinese information can be shown in ES environments. The realization process is described in Algorithm 1.

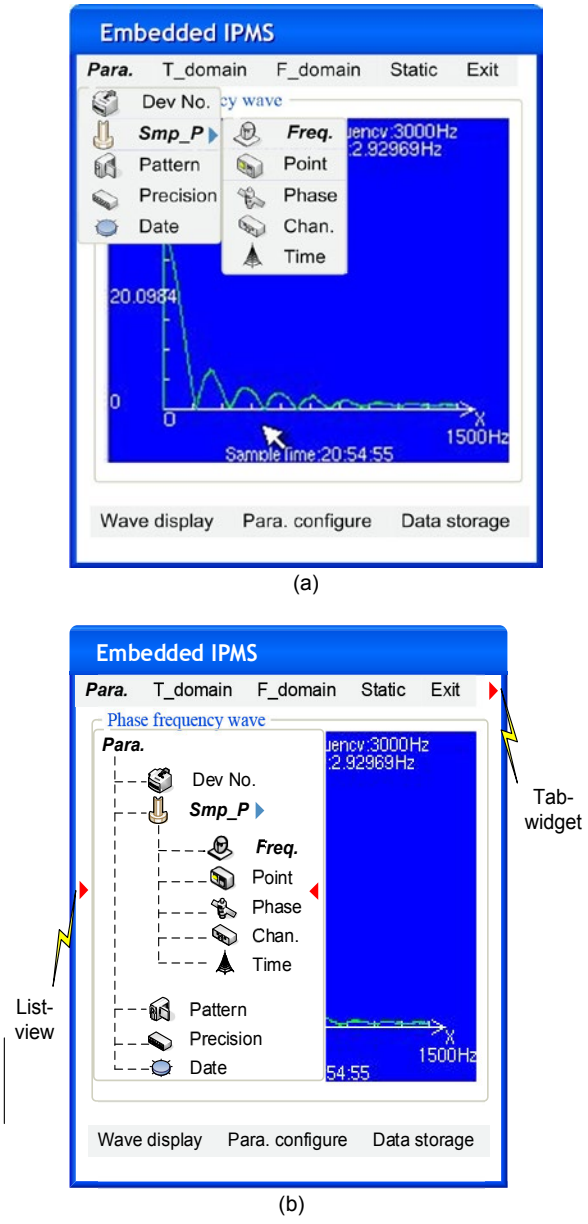

Fig. 11 Embedded IPMS interface with different object placement styles: (a) traditional menu-style object placement scheme; (b) optimized page placement scheme by the proposed scrolling page method

References to color refer to the online version of this figure

First, a pre-compiled header file that contains the segment variable definition of the compact Chinese library is created. The kernel codes are shown.

Algorithm 1 Chinese information show

```
1  #ifdef QTE_VERSION /*Define version*/
2      #include <qtextcodec.h>
3      #define GB(s) /*Define Chinese internal code*/
4      (QTextCodec::codecForName("GBK")->
           toUnicode(s));
5      #define AGB(s)
6      (QTextCodec::codecForName("GBK")->
           fromUnicode(s));
7  #else
8      #include <qgb18030codec.h>;
9      #define GB(s) /*Define standard Chinese character*/
10     (QGb18030Codec::codecForName("GB18030")->
           toUnicode(s))
11     #define AGB(s)
12     (QGb18030Codec::codecForName("GB18030")->
           fromUnicode(s))
13   #define SEG(MECH) /*Define the mechanic device*/
14 #endif
```

Then if the header file is cited in an embedded application program, the Chinese information will be shown in the embedded GUI. For example, the code to realize a radio-button named '停止采集系统' (stop the real-time data collection system) is as shown below:

showRadioButton1=new QRadioButton(GB(tr("停止采集系统")), setButtonGroup1, "showRadioButton1").

Of course, to show Chinese information in the embedded target system, the corresponding word library file should be downloaded into the flash memory of the target system. After compilation of Qt/E, there are large numbers of font files with the extended name of .qpf in the directory qte/lib/fonts. As far as the font files are concerned, it is not required to download all files into the target system. Only the file with the name unifont_160_50.qpf, i.e., the Unicode font library file that contains the compact Chinese library segment, needs to be downloaded into the target system. This method reduces the memory space of the Chinese library effectively and can satisfy the requirements of the embedded IPMS-GUI well (Fig. 12). Moreover, the method supports the transformation from GBK to Unicode. Accordingly, the Qt/E-based IMMS-GUI can also display the compact font libraries of original complex Chinese, Japanese, or Korean.

6 Embedded GUI application instance and simulated/industrial experiments

6.1 Embedded fixed instrument platform and cross-platform performance

The ES environment test is the indispensable work to verify the performance of the lightweight GUI library. As indicated in Section 1, an embedded IPMS acts commonly as a fixed EIS for one or several devices. To perform the test for a fixed EIS, we established an embedded fixed instrument experimental platform with a dual advanced RISC machine (ARM) architecture (Fig. 13).

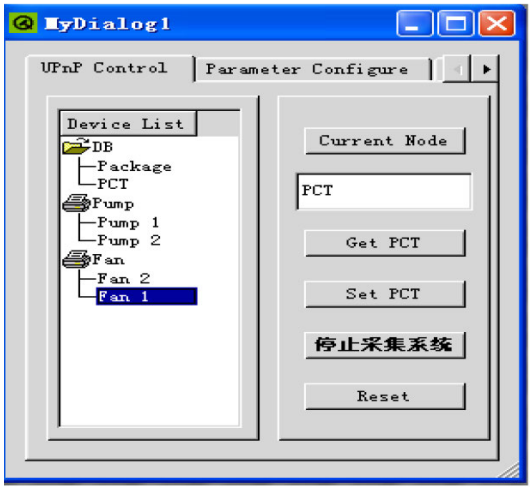

Fig. 12 Chinese information interface in embedded system environments

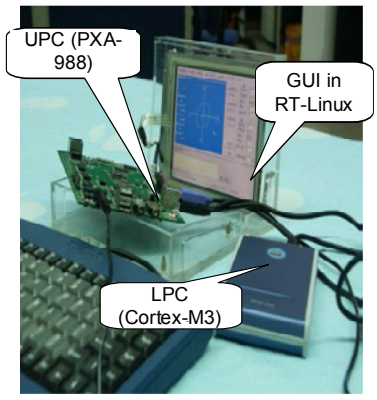

Fig. 13 Embedded fixed instrument platform with dual-ARM architecture

The fixed instrument experimental platform consists of two independent embedded computer

systems: an upper position computer (UPC) and a lower position computer (LPC). A UPC is a real-time data processing and management system based on PXA-988; an LPC is an embedded real-time data collection and pre-processing system based on TI-Cortex-M3. Referring to the self-designed data communication protocols, the monitoring data communication and control command interactions between a UPC and an LPC were performed over a universal serial bus (USB) port (Li et al., 2013).

For the experimental procedures, an embedded OS (RT-Linux) compatible with Qt/E was ported into a UPC, and then the shared library files of IPMS-GUI and the executable files of the embedded program were transmitted to the UPC. Then the testing experiments for interface switching speed and data algorithm processing time were iteratively executed. The results proved that the embedded IPMS application program runs stably, can display the real-time data graphics fluently, and performs the following tasks: parameter configuration, real-time data display, historical data playback, data storage, data query, file management, and power management.

Cross-platform performance is an important technical index for the embedded lightweight GUI component library. Accordingly, the cross-platform testing experiments that consider the transplantation and porting capacity of the functional components were executed on two classical ES environments: RT-Linux and Windows-CE. Under RT-Linux, an automatic coding tool (qmake) provided by Qt/E was adopted to create the compilation guide file (Makefile) for embedded program compilation. Because manual Makefile editing is complex and prone to cause errors, the qmake can directly create the Makefile for different platforms and compilers by a simple Qt/E project file (Dalheimer and Hansen, 2002).

In the Windows-CE framework, GUI components can be regarded as a part of the Qt/E plug-ins of embedded visual C++ (EVC++). In this way, it reduces the difficulty of application program development effectively. Application instance interfaces with 640×480 pixels of embedded IPMS in RT-Linux and Windows-CE are shown in Figs. 13 and 14. It can be stated that this component library has good cross-platform performance.

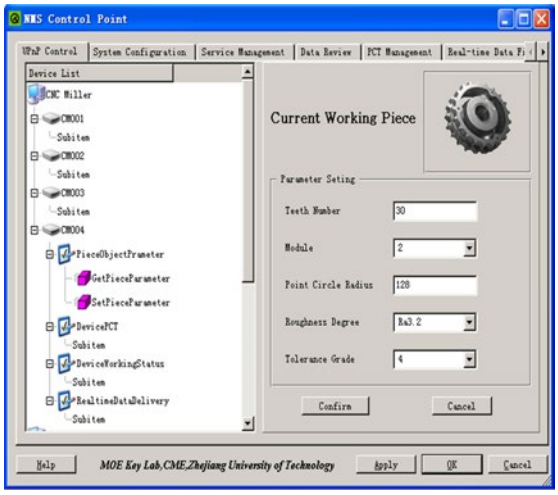

Fig. 14 Embedded IPMS interface developed by the component library in Windows-CE

6.2 Handheld mobile environments appraisal

With the development of modern industrial technologies, ubiquitous computing and wearable electronic appliances become the research hotspots in the information electronics area (Zheng et al., 2010a, 2010b; Park and Lee, 2011). Accordingly, embedded handheld mobile instruments are an important developing direction for IPMSs because of their flexibility and instantaneous performance. To address this question, we set up the corresponding handheld ES experimental platform oriented to the lightweight GUI component library (Fig. 15). The platform also adopts a dual-CPU architecture, where the UPC is a PDA (PocketPC) and the LPC is a real-time physical signal collection card based on TI-MSP430. According to the fundamental requirements of handheld ES environments, real-time data transformation and control information interaction were realized between UPC and LPC by the three-line zero-modem method.

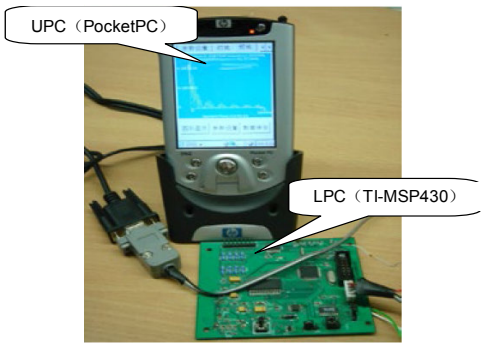

Fig. 15 Handheld ES environments platform

With regard to the embedded software environments, the open palmtop integrated environment (OPIE), a special handheld operating system based on the familiar-Linux kernel which can be compatible with Qt/E-2.3.7 or higher versions, was transplanted into PDA (UPC) first. Second, the lightweight GUI shared library files designed for handheld environments and the application program executable files were downloaded into the UPC using the serial or Ethernet port.

Subsequently, the performance test considering the interface switching speed and monitoring algorithm efficiency for handheld mobile environments was completed. The results proved that the mobile IPMS instance runs stably and can complete the functions of parameter configuration, device management, data storage, data query, and file management; i.e., parallel multi-channel real-time data processing can be realized.

Similarly, the handheld cross-platform performance of the embedded GUI library was checked against OPIE (Qtopia) and Windows-CE (PocketPC-2003). Qtopia is a special Qt/E version designed for handheld ES platforms. It possesses a more compact architecture and smaller memory storage space, and can provide a lightweight GUI framework with 240×320 pixels.

Based on the embedded GUI library, two mobile IPMS instances were developed on Qtopia and PocketPC-2003 (Figs. 16 and 17). Results proved that the embedded GUI library is adequate for handheld intelligent instruments and has better transplantation performance, compared with Micro-Windows, Xfree86, MiniGUI, Windows-CE-GUI, and Android-GUI.

6.3 Simulations and discussion

6.3.1 Case study of the signal-slot communication mechanism

To verify the effectiveness of the proposed signal-slot communication mechanism for industry monitoring, we provided a numerical case in which four object instances are included. The message connection relationship is shown in Fig. 18. The four instances contain six signals and seven slots. The corresponding connection arrangement list is shown in Table 1, where the signals and slots of one object cannot be allowed to connect to each other. There are 31 types of connections; for example, C1143 stands for the connection (Object-1, Signal-1, Object-4, Slot-3).

With regard to the case in Fig. 18, the comparative experiments with the message mapping mechanism of Windows-CE-GUI and Android-GUI have been performed. The experiment is oriented to a large-quantity data read-write process on an embedded flash memory, which has a high possibility to create errors, especially for audio/video data. The experimental route is described as follows: (1) Two groups of audio/video data sequences were randomly created, of which each group includes 10 data packets; (2) The data packets were written into the memory, and then read to the user buffer based on the pointer of the starting address; (3) The signal (message) was a single-click event, and the slot (mapping) functions of

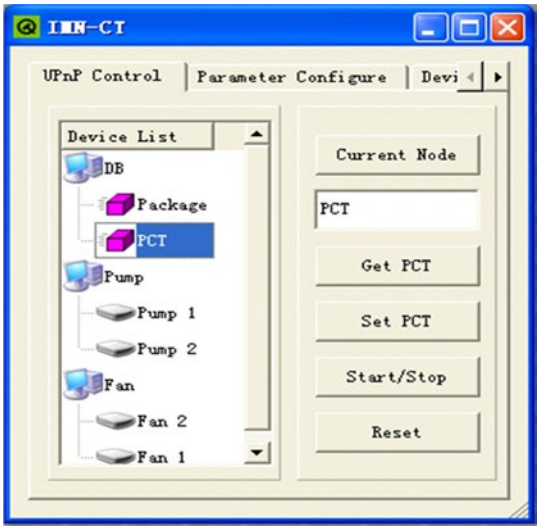

Fig. 16 Handheld IPMS interface on Qtopia (OPIE)

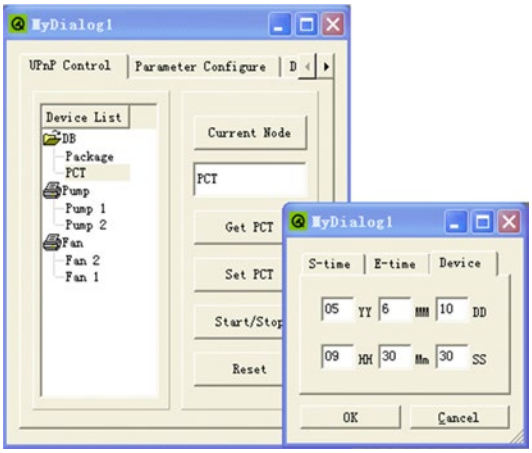

Fig. 17 Handheld mobile IPMS interface on PocketPC-2003 (Windows-CE)

Table 1 Connection arrangement list of the four objects

Slot		Object-1		Object-2	Object-3	Object-4	
		Signal-1	Signal-2	Signal-1	Signal-1	Signal-1	Signal-2
Object-1	Slot-1	–	–	C2111	C3111	C4111	C4211
Object-2	Slot-1	C1121	C1221	–	C3121	C4121	C4221
	Slot-2	C1122	C1222	–	C3122	C4122	C4222
Object-3	Slot-1	C1131	C1231	C2131	–	C4131	C4231
Object-4	Slot-1	C1141	C1241	C2141	C3141	–	–
	Slot-2	C1142	C1242	C2142	C3142	–	–
	Slot-3	C1143	C1243	C2143	C3143	–	–

the 31 types of connections executed the read-write process of 20 data packets; (4) Experiments were performed 1000 times. If all the read-write processes were finished, the number of correct read-write times was recorded as the result and the numbers of correct ratios of the three GUIs were obtained.

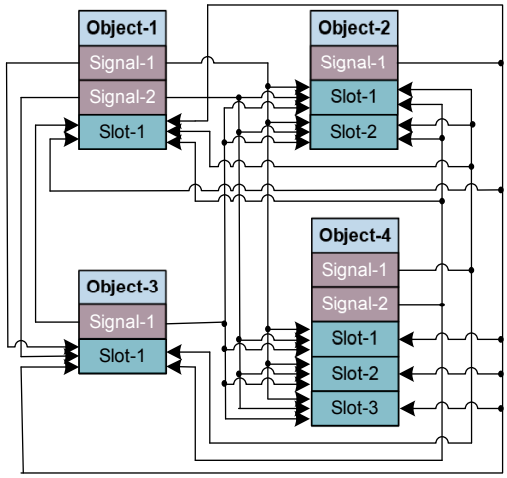

Fig. 18 Signal-slot communication connection of four simulated objects

The comparative experiment results for audio data sequences are shown in Fig. 19. From the figure, the following conclusions can be obtained: (1) With the increase in data quantity, the read-write correct ratio decreases accordingly. The average correct ratios of the proposed method, Windows-CE-GUI, and Android-GUI are 92.1%, 78.2%, and 73.5%, respectively. (2) The proposed signal-slot mechanism has better security performance than Windows-CE-GUI and Android-GUI, and Android-GUI is with the worst correct ratio. (3) The proposed method is subjected to lower influences caused by the variations of data quantity, and the average correct ratio can be higher than 92%. However, the ratios of the other two methods decrease apparently by the increment of data quantity; for the data packets of 1000 KB, their correct ratios have decreased to less than 60%.

The comparative experiment results for video data sequences are shown in Fig. 20, where the data quantity increases to 100 times the audio data quantity. We can obtain the variation regulars as follows: (1) With the remarkable increment of data quantity, the read-write correct ratios of the three methods decline continuously. The average correct ratios of the proposed method, Windows-CE-GUI, and Android-GUI are 82.4%, 61.2%, and 61.8%, respectively. (2) The proposed method has still the highest correct ratio of more than 82%. (3) For the video data packets, the total performance of Android-GUI is better than

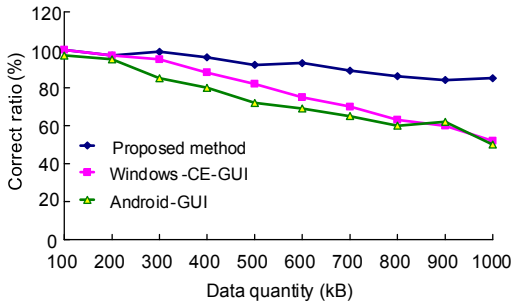

Fig. 19 Audio data read-write experiment results

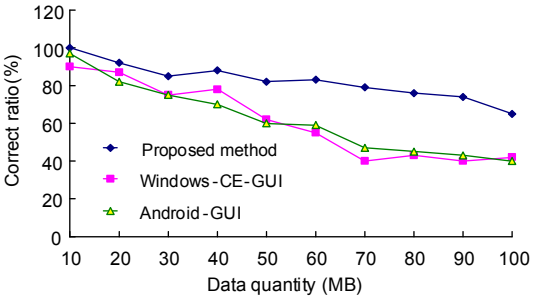

Fig. 20 Video data read-write experiment results

that of Windows-CE-GUI. (4) When the data quantity is larger than 50 MB, the read-write correct ratios of the two methods are about 60% and hard to match the practical requirements of industry monitoring.

6.3.2 Case study of the proposed ergonomics optimization method

To check the advantages of the proposed ergonomics optimization method, an operational test was carried out by the industrial psychological method, in which 100 users (20 teachers and 80 students) were selected as the test objects.

The involved ergonomics indexes are described as follows (Du, 2008): (1) The total evaluation score (TES) is the sum of the order values of questionnaire items. It is a subjective data index, which can represent the total impression of users and reflect the friendliness of the GUI. (2) System study time (SST) is the time period from the moment at which users begin to contact the GUI to the moment at which they are familiar with all the functional interfaces. SST can characterize the performance of visual object placement. (3) Specified task operating time (STOT) is a special data index for an IPMS, recording the time length to perform a monitoring task, such as the time-domain task, the frequency-domain task, and the data storage task. It can characterize the availability of GUI. (4) Normalized searching performance (NSP) records the time scores to search a functional interface or a configurable dialog. It can represent the accessibility performance. (5) Normalized adapting performance (NAP) describes the adaptation time for users with different educational backgrounds and can characterize the applicability performance. (6) Normalized understanding performance (NUP) is related to the understanding process for the labels of all interfaces and parameters. It can characterize the suitability for different users. (7) Normalized consistency of display and operation (NCDO) is also a special data index for an IPMS, describing the switching consistency of different sessions.

All the users should touch GUI at the first time and complete the operation of setting the sampling frequency. The technical factors that can affect the operating performance of the GUI are first defined. Then the fuzzy evaluation method is adopted to obtain the experimental results (Table 2).

The following results can be obtained. Compared with the menu style, the scrolling page method can increase the TES by 26.8%. This illustrates that the proposed method can make GUI friendly and apt to form an evolving relation between users and computers. SST, STOT, and NSP are improved apparently, proving that the proposed method is with better visual performance, availability, and accessibility, respectively. NAP and NUP increase by more than 20.8% and 26.7%, respectively, illustrating that the proposed method is conducive to the combination of human and computer. The applicability performances for users with different educational levels are optimized. Regarding the embedded IPMS-GUI, NCDO is one of the most important indexes for industry monitoring, in which NCDO is increased by 42.2% and the switching consistency of different sessions is improved. The above operation is performed by the multi-thread task scheduling method indicated in Section 4.2.

6.4 Industrial experiments and discussion

6.4.1 Fundamental performance comparison

To check the working performance of the improved Qt/E-based IPMS-GUI, we performed the comparative experiments with Mini-GUI and Windows-CE-GUI. The experimental results are shown in Table 3. From the table, the following conclusions can be obtained: (1) Compared with Mini-GUI and Windows-CE-GUI, the proposed GUI library is with apparent superiority for memory space. Containing the monitoring algorithms and Chinese fonts, the memory space can be tailored to 900 KB. (2) By the multi-thread scheduling method, IPMS-GUI has the same parallel task processing ability as Windows-CE-GUI and better ability than Mini-GUI obviously. (3) Owing to the signal-slot communication mechanism, IPMS-GUI can obtain the best session switching performance, compared with the other two GUIs. Moreover, all the library files are with pure text format and are coded by the basic C/C++. In combination with the merit on transportability, IPMS-GUI can have the best cross-platform performance.

6.4.2 Industrial field experiments and user appraisals

With regard to IPMS-GUI, the industrial field experiments are indispensable procedures and the final targets. To address the issue, five groups of industrial experiments have been performed in four

large-scale enterprises: Wuhan Iron and Steel Corporation (WISCO), Bao Steel Group (BSG), Shengli Oil Field (SLOF), and Shenyang Pump Factory (SPF). The experimental scenarios are shown in Fig. 21.

Fig. 21 Industrial field experiment scenarios in four large-scale enterprises

The experiments were divided into five groups: three fixed IPMS experiments in WISCO, SLOF, and BSG, and two handheld IPMS experiments in SLOF and SPF (Fig. 21). From the user appraisement results shown in Table 4, the following conclusions can be obtained:

1. The embedded IPMS-GUI achieves good user appraisals from the four enterprises. The average satisfaction ratio can be higher than 90%.

2. The effects of the handheld GUI are better than those of the fixed one, and the handheld experiment in SPF obtains the best scores, where the TES value is 3.83.

3. The average value of the four indexes of ergonomics can be higher than 0.85, which proves that the proposed method can satisfy the practical human-machine requirements for industry monitoring.

7 Conclusions

The development of an embedded lightweight GUI has become an indispensable technology of modern industry monitoring. To solve the problems of current GUI implementation methods, such as large use of memory space, bad configurability, and poor real-time performance, an embedded lightweight IPMS-GUI component library design method based on Qt/E has been proposed. For the above research target, the corresponding research has been performed. The main conclusions of this study are as follows.

1. An E-R model oriented to an embedded IPMS-GUI component library has been built up to define the functional modules' framework and data coupling relations, which can provide modularization rules for the development of the embedded IPMS-GUI.

Table 2 Operating performance comparative experiment results of menu-style and list-tab style

Style	TES	SST (s)	STOT (s)	NSP	NAP	NUP	NCDO
Menu style	2.87	178	5	0.68	0.77	0.75	0.67
The proposed method	3.64	125	3	0.82	0.93	0.95	0.94

Table 3 Working performance comparative experiment results with Mini-GUI and Windows-CE-GUI

GUI	Least library space (KB)	Maximum parallel process number	Average session switch time (s)	Number of compatible software platforms	Number of compatible hardware platforms
Mini-GUI	1024	8	0.82	12	8
Windows-CE-GUI	>1024	32	0.53	1	8
IPMS-GUI	900	32	0.55	18	12

Table 4 User appraisement results of industrial field experiments

Index	SPF-handheld	SLOF-handheld	WISCO-fixed	SLOF-fixed	BSG-fixed
NSP	0.94	0.92	0.85	0.88	0.87
NAP	0.94	0.90	0.90	0.82	0.80
NUP	0.98	0.96	0.95	0.93	0.90
NCDO	0.97	0.91	0.88	0.90	0.85
TES	3.83	3.69	3.58	3.53	3.42
Satisfaction ratio	95%	97%	90%	93%	87%

2. To adapt to different embedded computer architectures, a cross-compilation environment platform that can hold multiple embedded targets has been constructed. A virtual QVFB platform was developed to improve the efficiency of GUI development.

3. The Qt/E shared library files have been tailored to 640 KB. The self-designed monitoring algorithms were embedded into the GUI library and could be compressed to 200 KB. The total memory use of the GUI library, including the Chinese fonts, was less than 900 KB and could satisfy the running requirements of currently popular embedded target environments.

4. Using a multi-thread method, a status transmission model for distributed real-time tasks was set up. Based on the model, the parallel digital signal processing capabilities of IPMS-GUI were enhanced, NCDO was improved, and the real-time performance and robustness were guaranteed.

5. By using the Qt/E-based signal-slot communication interfaces, a message mapping mechanism that does not require a call-back pointer has been realized, and the context switching speed was improved. A case study proved that the signal-slot mechanism can increase real-time read-write correction ratios by more than 26% and 29%, compared with Windows-CE-GUI and Android-GUI, respectively.

6. The human-computer interaction process has been optimized by a scrolling page method and the ergonomics performance has been strengthened. A compact Chinese library with 10 specialty segments has been proposed and the display of Chinese information on the embedded IPMS-GUI was realized.

7. A universal embedded GUI shared component library oriented to IPMSs has been developed. The IPMS-GUI instance testing on a fixed instrument platform and a handheld instrument platform proved that the embedded GUI library can satisfy the fundamental requirements of an IPMS and has better cross-platform performance, compared with Micro-Windows, Xfree86, MiniGUI, Windows-CE-GUI, and Android-GUI.

8. The performance of comparative experiments with Mini-GUI and Windows-CE-GUI proved that the improved Qt/E-based IPMS-GUI has better working performance compared with Mini-GUI, and the session switching time is close to that of Windows-CE-GUI.

9. Industrial field experiments in four large-scale enterprises proved that the embedded IPMS-GUI achieves good user appraisals. The average satisfaction ratio can be more than 90% and the effects of the handheld GUI are better than those of the fixed GUI.

Generally, we have proposed an embedded lightweight GUI library design method oriented to IPMSs. We hope our study can provide useful references to embedded human-computer interface theories and industrial practices, and supply experience evidence and technical supports for the engineering areas of component-based software design, embedded application software development, condition-based monitoring technologies, handheld inspection instruments, and system ergonomics optimization. Future research will be executed around the facets of the hardware layer control method, real-time performance of monitoring algorithms, and ergonomics performance optimization.

References

Acciani G, Fornarelli G, Giaquinto A, 2011. A fuzzy method for global quality index evaluation of solder joints in surface mount technology. *IEEE Trans Ind Inform*, 7(1):115-124. https://doi.org/10.1109/TII.2010.2076292

Ahn SH, Sul D, Choi SH, et al., 2006. Implementation of lightweight graphic library builder for embedded system. IEEE Int Conf on Advanced Communication Technology, p.166-168. https://doi.org/10.1109/ICACT.2006.205944

Barrero F, Toral S, Vargas M, et al., 2010. Internet in the development of future road-traffic control systems. *Internet Res*, 20(2):154-168. https://doi.org/10.1108/10662241011032227

Cecotti H, 2016. A multimodal gaze-controlled virtual keyboard. *IEEE Trans Hum-Mach Syst*, 46(4):601-606. https://doi.org/10.1109/THMS.2016.2537749

Chen ST, Tan DP, 2018. A SA-ANN-based modeling method for human cognition mechanism and the PSACO cognition algorithm. *Complexity*, 2018:6264124. https://doi.org/10.1155/2018/6264124

Chevalier A, Kicka M, 2006. Web designers and web users: influence of the ergonomic quality of the web site on the information search. *Int J Hum-Comput Stud*, 64(10): 1031-1048. https://doi.org/10.1016/j.ijhcs.2006.06.002

Dalheimer MK, Hansen S, 2002. Embedded systems: embedded development with qt/embedded. *Dr Dobbs J*, 27(3):48-54.

Drossu R, Obradovic Z, Fletcher J, 1996. A flexible graphical user interface for embedding heterogeneous neural network simulators. *IEEE Trans Edu*, 39(3):367-374. https://doi.org/10.1109/13.538760

Du F, 2008. GUI Design Based on Ergonomics. MS Thesis, Nanjing University of Aeronautics and Astronautics, Nanjing, China (in Chinese).

Ji SM, Xiao FQ, Tan DP, 2010. Analytical method for softness abrasive flow field based on discrete phase model. *Sci China Technol Sci*, 53(10):2867-2877. https://doi.org/10.1007/s11431-010-4046-9

Ji SM, Weng XX, Tan DP, 2012. Analytical method of softness abrasive two-phase flow field based on 2D model of LSM. *Acta Phys Sin*, 61(1):010205.

Ji SM, Ge JQ, Tan DP, 2017. Wall contact effects of particle-wall collision process in a two-phase particle fluid. *J Zhejiang Univ-Sci A* (*Appl Phys & Eng*), 18(12):958-973. https://doi.org/10.1631/jzus.A1700039

Jin F, Wu ZH, 2008. Lightweight graphics device driver and graphical user interface based on embedded Linux. *Trans Beijing Inst Technol*, 28(3):233-236. https://doi.org/10.15918/j.tbit1001-0645.2008.03.018

Li C, Ji SM, Tan DP, 2012. Study on machinability and the wall region of solid-liquid two phase softness abrasive flow. *Int J Adv Manuf Technol*, 61(9-12):975-987. https://doi.org/10.1007/s00170-011-3621-y

Li C, Ji SM, Tan DP, 2013. Multiple-loop digital control method for 400Hz inverter system based on phase feedback. *IEEE Trans Power Electron*, 28(1):408-417. https://doi.org/10.1109/TPEL.2012.2188043

Li J, Ji SM, Tan DP, 2017. Improved soft abrasive flow finishing method based on turbulent kinetic energy enhancing. *Chin J Mech Eng*, 30(2):301-309. https://doi.org/10.1007/s10033-017-0071-y

Li X, Horie M, Kagawa T, 2014. Pressure-distribution methods for estimating lifting force of a swirl gripper. *IEEE/ASME Trans Mechatron*, 19(2):707-718. https://doi.org/10.1109/TMECH.2013.2256793

Li X, Li N, Tao GL, 2015. Experimental comparison of Bernoulli gripper and vortex gripper. *Int J Prec Eng Manuf*, 16(10):2081-2090. https://doi.org/10.1007/s12541-015-0270-3

Liao YX, Li X, Zhong W, et al., 2016. Study of pressure drop-flow rate and flow resistance characteristics of heated porous materials under local thermal non-equilibrium conditions. *Int J Heat Mass Transf*, 102:528-543. https://doi.org/10.1016/j.ijheatmasstransfer.2016.05.101

Lin ZS, Yu SM, Lu JH, 2015. Design and ARM-embedded implementation of a chaotic map-based real-time secure video communication system. *IEEE Trans Circ Syst Video Technol*, 25(7):1203-1216. https://doi.org/10.1109/TCSVT.2014.2369711

Mazzei D, Vozzi F, Cisternino A, et al., 2008. A high-throughput bioreactor system for simulating physiological environments. *IEEE Trans Ind Electron*, 55(9):3273-3280. https://doi.org/10.1109/TIE.2008.928122

Park J, Lee J, 2011. A beacon color code scheduling for the localization of multiple robots. *IEEE Trans Ind Inform*, 7(3):467-475. https://doi.org/10.1109/TII.2011.2158833

Ramos MA, Penteado RAD, 2008. Embedded software revitalization through component mining and software product line techniques. *J Univ Comput Sci*, 14(8):1207-1227. https://doi.org/10.3217/jucs-014-08-1211

Rehault F, 2010. Windows mobile advanced forensics: an alternative to existing tools. *Dig Invest*, 7(1-2):38-47. https://doi.org/10.1016/j.diin.2010.08.003

Riskedal E, 2008. Qt and Windows CE. *Dr Dobbs J*, 33(6): 30-45.

Saponara S, Petri E, Fanucci L, et al., 2011. Sensor modeling, low-complexity fusion algorithms, and mixed-signal IC prototyping for gas measures in low-emission vehicles. *IEEE Trans Instrum Meas*, 60(2):372-384. https://doi.org/10.1109/TIM.2010.2084230

Steblovnik K, Zazula D, 2011. A novel agent-based concept of household appliances. *J Intell Manuf*, 22(1):73-88. https://doi.org/10.1007/s10845-009-0279-5

Su LJ, Zheng NG, Yao M, et al., 2014. A computational model of the hybrid bio-machine MPMS for ratbots navigation. *IEEE Intell Syst*, 29(6):5-13. https://doi.org/10.1109/MIS.2014.91

Tan DP, Zhang LB, 2014. A WP-based nonlinear vibration sensing method for invisible liquid steel slag detection. *Sensor Actuat B Chem*, 202:1257-1269. https://doi.org/10.1016/j.snb.2014.06.014

Tan DP, Ji SM, Li PY, et al., 2010. Development of vibration style ladle slag detection method and the key technologies. *Sci China Technol Sci*, 53(9):2378-2387. https://doi.org/10.1007/s11431-010-4073-6

Tan DP, Ji SM, Jin MS, 2013a. Intelligent computer-aided instruction modeling and a method to optimize study strategies for parallel robot instruction. *IEEE Trans Edu*, 56(3):268-273. https://doi.org/10.1109/TE.2012.2212707

Tan DP, Li PY, Ji YX, et al., 2013b. SA-ANN-based slag carry-over detection method and the embedded WME platform. *IEEE Trans Ind Electron*, 60(10):4702-4713. https://doi.org/10.1109/TIE.2012.2213559

Tan DP, Ji SM, Fu YZ, 2016a. An improved soft abrasive flow finishing method based on fluid collision theory. *Int J Adv Manuf Technol*, 85(5-8):1261-1274. https://doi.org/10.1007/s00170-015-8044-8

Tan DP, Yang T, Zhao J, et al., 2016b. Free sink vortex Ekman suction-extraction evolution mechanism. *Acta Phys Sin*, 65(5):054701. https://doi.org/10.7498/aps.65.054701

Tan DP, Zhang LB, Ai QL, 2016c. An embedded self-adapting network service framework for networked manufacturing system. *J Intell Manuf*, in press. https://doi.org/10.1007/s10845-016-1265-3

Tan DP, Li L, Zhu YL, et al., 2017a. An embedded cloud database service method for distributed industry monitoring. *IEEE Trans Ind Inform*, in press. https:// doi.org/10.1109/TII.2017.2773644

Tan DP, Ni YS, Zhang LB, 2017b. Two-phase sink vortex suction mechanism and penetration dynamic characteristics in ladle teeming process. *J Iron Steel Res Int*, 24(7): 669-677. https://doi.org/10.1016/S1006-706X(17)30101-2

Veltcheva AD, Soares CG, 2012. Analysis of abnormal wave groups in Hurricane Camille by the Hilbert Huang transform method. *Ocean Eng*, 42:102-111. https://doi.org/10.1016/j.oceaneng.2011.12.013

Wang J, Li DJ, Yang CJ, et al., 2015. Developing a power monitoring and protection system for the junction boxes of an experimental seafloor observatory network. *Front Inform Technol Electron Eng*, 16(12):1034-1045. https://doi.org/10.1631/FITEE.1500099

Wu ZH, Zheng NG, Zhang SW, et al., 2016. Maze learning by a hybrid brain-computer system. *Sci Rep*, 6:31746. https://doi.org/10.1038/srep31746

Wulf V, Pipek V, Won M, 2008. Component-based tailorability: enabling highly flexible software applications. *Int J Hum Comput Stud*, 66(1):1-22. https://doi.org/10.1016/j.ijhcs.2007.08.007

Xu LD, Viriyasitavat W, Ruchikachorn P, et al., 2012. Using propositional logic for requirements verification of service workflow. *IEEE Trans Ind Inform*, 8(3):639-646. https://doi.org/10.1109/TII.2012.2187908

Yao MQ, Yang K, Xu CY, et al., 2015. Design of a novel RTD-based three-variable universal logic gate. *Front Inform Technol Electron Eng*, 16(8):694-699. https://doi.org/10.1631/FITEE.1500102

Yin S, Wang G, Gao H, 2015. Data-driven process monitoring based on modified orthogonal projections to latent structures. *IEEE Trans Contr Syst Technol*, 24(4):1480-1487. https://doi.org/10.1109/TCST.2015.2481318

Zeng X, Ji SM, Tan DP, et al., 2013. Softness consolidation abrasives material removal characteristic oriented to laser hardening surface. *Int J Adv Manuf Technol*, 69(9-12): 2323-2332. https://doi.org/10.1007/s00170-013-4985-y

Zeng X, Ji SM, Jin MS, et al., 2016. Research on dynamic characteristic of softness consolidation abrasives in machining process. *Int J Adv Manuf Technol*, 82(5-8):1115-1125. https://doi.org/10.1007/s00170-015-7392-8

Zhang K, Kang JU, 2011. Real-time numerical dispersion compensation using graphics processing unit for Fourier-domain optical coherence tomography. *Electron Lett*, 47(5):309-310. https://doi.org/10.1049/el.2011.0065

Zhang M, Jiang JZ, Liu CH, 2013. Development of a multi-function gateway node oriented environment monitoring in greenhouse. *Sens Lett*, 11(6-7):1236-1239. https://doi.org/10.1166/sl.2013.2852

Zheng NG, Wu Z, Lin M, et al., 2010a. Enhancing battery efficiency for pervasive health-monitoring systems based on electronic textiles. *IEEE Trans Inform Technol Biomed*, 14(2):350-359. https://doi.org/10.1109/TITB.2009.2034972

Zheng NG, Wu ZH, Lin M, et al., 2010b. Infrastructure and reliability analysis of electric networks for E-textiles. *IEEE Trans Syst Man Cybern Part C*, 40(1):36-51. https://doi.org/10.1109/TSMCC.2009.2031497

Zheng NG, Su LJ, Zhang DQ, et al., 2015. A computational model for ratbot locomotion based on cyborg intelligence. *Neurocomputing*, 170(C):92-97. https://doi.org/10.1016/j.neucom.2014.12.115

Zhou HJ, Xiang R, 2013. MicroWindows-based multi-device support intelligent Chinese input system. *J Comput Appl*, 33(7):2067-2070. https://doi.org/10.11772/j.issn.1001-9081.2013.07.2067

Zhuo XF, Fan JB, Chen B, 2002. Application of Linux multilineality in GUI programming. *J Southwest Univ Sci Technol*, 17(3):21-24.