# Middleware Software for Embedded Systems

Yang-Hsin Fan
Dept. of Computer Science and Information Eng.
National Taitung University
Taitung City, Taiwan 95002
yhfan@nttu.edu.tw

Jan-Ou Wu
Dept. of Electronic Eng.
De Lin Institute of Technology
Tu-Cheng, Taipei City, Taiwan 236
janou@ms42.hinet.net

***Abstract*—Embedded systems with communicating and computing ability and multimedia functions work to every corner of daily life. However, the diverse architectures of embedded systems cause problems corresponding to reuse, portability and dependability. Middleware is a set of software that executes between operating system and application to solve stated problems. The advantages include unified interface, scalable and transparent abilities. This project investigates middleware technology on embedded systems and then proposes embedded middleware architecture to overcome the problems relating to reuse, portability, dependability and transparency. The proposed middleware consists of API module, service manager module and content manager module. The API module is designed to communicate with upper layer and lower layer by unified interface. Also, it provides a set of methods for solving problems of portability and dependability. The service manager module conducts a set of services for application. Moreover, it is able to automatically generate and deploy content information. The content manager module organizes and supplies information to the service manager module. Additionally, it solves transparent problem by ASCII and URL representation. In order to verify the feasibility for the proposed approach, we design the application of touch screen for embedded middleware system. Not only middleware but also embedded operating system, shell script, embedded graphical user interface and application are validated. Finally, we assess the functionality and integrated test of embedded middleware system by S3C2410 and XScale PXA270 system.**

***Keywords-middleware software, embedded software, embedded system.***

## I. INTRODUCTION

Embedded systems serve the functionalities spread over information, communication as well as multimedia. Recently, the benefits of embedded systems such as smaller, smarter, portable characteristic leads it to greatly apply on *computer*, *communication*, *consumer and car* (4C). On the other hand, cloud computing and smart phone are generally regard as embedded system result in working to every corner of daily life. However, the diverse architectures of embedded systems cause problems corresponding to reuse, portability and dependability. Moreover, the communication and interchange issues on levels such as operating system versus to libraries, libraries versus to applications and system to system are significant challenges.

Owing to the architecture of embedded systems is diversity, each design of hardware and software systems are generally great time consuming. The challenges include interchange and communication for each levels or systems or devices. In 2007, Rincón *et al.* [1] discussed system level for communicating on hardware/software for embedded systems. They adopted distributed components platform [2]-[5] such as *common object request broker architecture* (CORBA) or Java *remote method invocation* (RMI) to solve flexibility and reuse issues for embedded systems. Unify model was also chosen to describe the components of hardware/software. Then, separately developing functionalities and communications of components achieves system specification. Finally, heterogeneity and scalability topics are stated to be overcome via distributed components model.

## II. PRELIMINARIES

*Network-centric middleware for group communication and resource sharing across heterogeneous embedded systems* (MORE) [6]-[8] is a middleware project that proposed by Europeans. This research aims heterogeneous embedded systems to discuss. This project based on *service oriented architecture* (SOA) [9][10] as well as adopted connector on communication for heterogeneous embedded systems. It presented $\mu$SOA approach to decrease the length of information and *extensible markup language* (XML) message. Figure 1 demonstrates the architecture of MORE. It divides embedded system into application layer, MORE middleware layer, operating system layer and Hardware layer. Three approaches are significant. One is SOA-based technology that is used on MORE middleware layer. Another is enabling services technique that plays a role of communication and interchange between operating system layer and Hardware layer. The other is $\mu$SOA approach that is used to communicate and exchange among heterogeneous embedded systems.

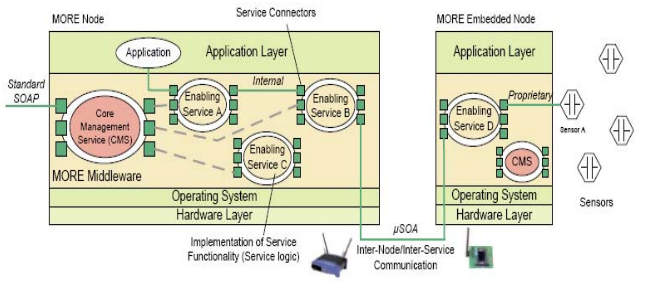

Figure 1. MORE architecture

Embedded software of embedded system addressed by Capra *et al.* [11] which applied on mobile embedded systems. Traditional middleware system such as *message-oriented middleware* (MOM) [12], *transaction-oriented middleware* (TOM) [13] and *object-oriented middleware* (OOM) [14][15]. They apply encapsulation technology to middleware system. Based on those techniques, the drawback is shield context information and behavior decision of the application which must be shield. Therefore, Li *et al*. presented middleware system for message-passing, remote method call, transaction and component services. Moreover, mobile middleware system must provide services which comprise of network operation system, collect setting context information and making adaptive decisions.

## III. MIDDLEWARE SOFTWARE

Traditional architecture of embedded systems comprise of six modules that demonstrates in Figure 2(a). From bottom-up of view, these modules are hardware module, peripheral driver module, *operating system* (OS) kernel module, file system module, libraries module and application module. Based on traditional embedded systems, we investigate the drawbacks and concluded as follow: 1) the reuse issue for diverse components; 2) the transparent subject for information interchange; 3) the portability topic for various applications; 4) the dependence for cross platforms; 5) the flexibility issue while developing systems; 6) the scalability issue. This study proposed middleware architecture for embedded system to combat those challenges. Figure 2(b) presents the proposed architecture of embedded system with middleware software that consists of four modules. From bottom-up of view, the modules are hardware, embedded operating system, middleware software as well as application. Inside embedded operating system, there are five sub-modules that include peripheral driver, OS kernel, file system, shell script, embedded *graphical user interface* (GUI).

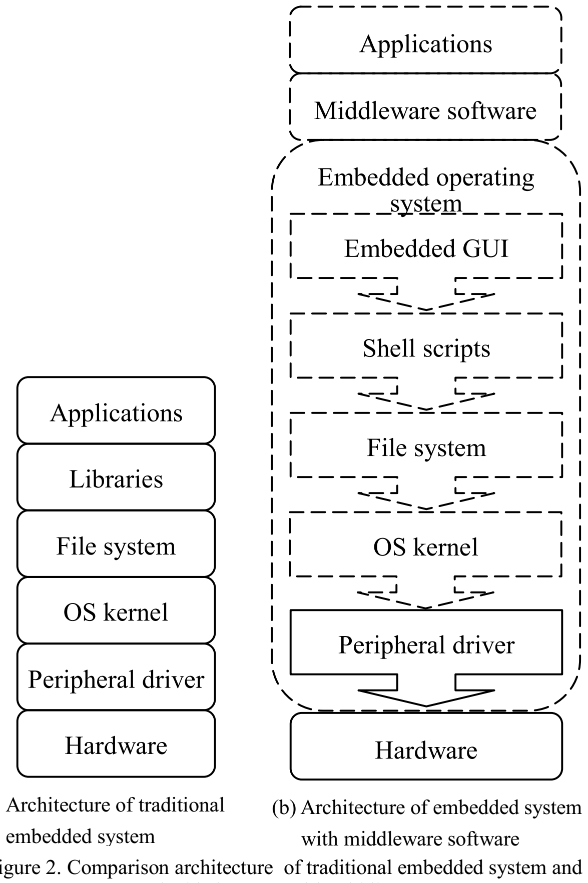

Figure 2. Comparison architecture of traditional embedded system and embedded system with middleware.

### *A. Architecture of embedded system with middleware software*

Not only embedded operating system but also middleware software is divide into sub-modules. Figure 3 illustrates the architecture of middleware software. The middleware software divides into three sub-modules that comprise middleware application programmable interface, service manager as well as content manager. The middleware application programmable interface sub-module provides unified interface for upper module (*i.e.* application module). On the other hand, it supplies some methods such encode, decode or file function, *etc.* for lower sub-module inside middleware software. Furthermore, it designs for solving for portability and dependability issues via a set of application programmable interfaces. Another sub-module namely service manager can automatic generation and deployment content for requirement of application. This sub-module also aim flexibility topic to provide related content for various applications. Moreover, it can automatic parse the structure of contents. Also, the verification and completion for content are verified in service manager sub-module. Furthermore, it achieves scalability via auto deployment content. In short, this sub-module designs for conquering flexibility and scalability for various embedded systems. The other sub-module called content manager develops for conducting content information. We design the format of content information for the purpose of solving transparent issue. The characteristics of content information include structural, self-description and readable text and simple syntax. Figure 4 displays the format of content information that consists of header, tags, elements, attributes and contents.

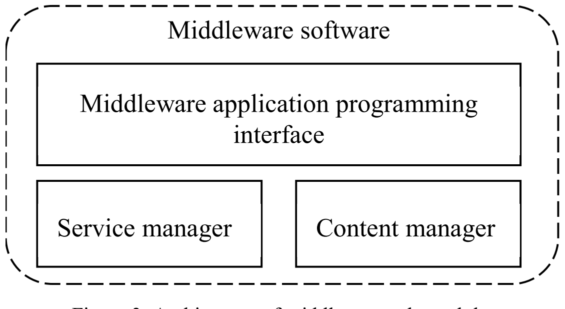

Figure 3. Architecture of middleware sub-modules.

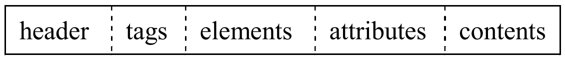

Figure 4. Content information format.

## B. *Design flow of middleware software*

Figure 5 shows the design flow of middleware software. First, we compile the embedded kernel for attaining embedded operating system. In step 2, we produce the embedded file system by compiling the root file system. In step 3, embedded kernel, file system and driver are verified via porting procedure to embedded platform. In step 4, the embedded *graphical user interface* (GUI) gives an interactive environment to user. It is more familiar to user than embedded text environment. On the other hand, the libraries of embedded GUI are provided in step 5 and 6 while designers develop embedded GUI applications. In step 7, we develop middleware software that comprises of service manager and content manager sub-modules. The middleware application programmable interfaces (APIs) are done in step 8. The proposed of middleware software are verified by two embedded systems. Those applications are designed in step 9. Scheduling applications procedures are deal on step 10. Finally, the whole embedded systems with middleware software are verified in step 11.

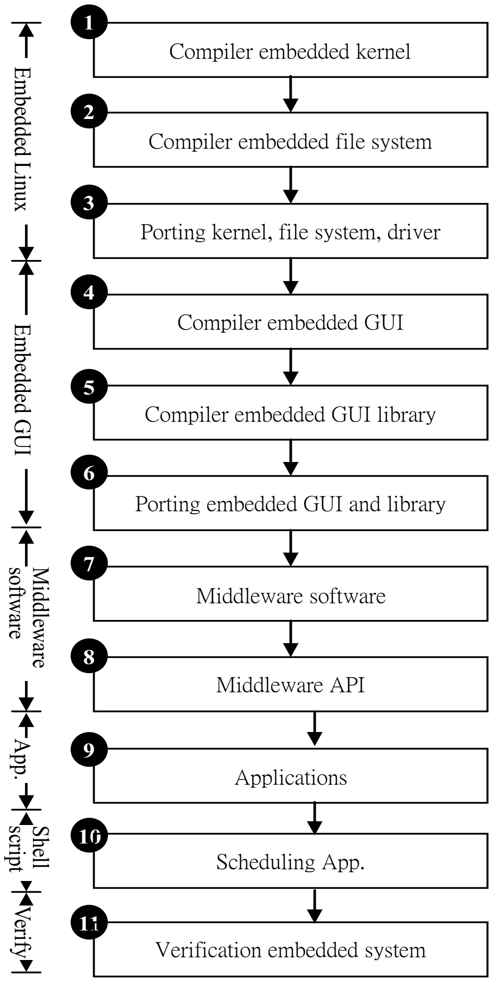
Embedded Linux
Embedded GUI
Middleware software
App.
Shell script
Verify
1 Compiler embedded kernel
2 Compiler embedded file system
3 Porting kernel, file system, driver
4 Compiler embedded GUI
5 Compiler embedded GUI library
6 Porting embedded GUI and library
7 Middleware software
8 Middleware API
9 Applications
10 Scheduling App.
11 Verification embedded system

Figure 5. Design flow of embedded system with middleware.

## IV. EXPERIMENTAL RESULTS

Figure 6 demonstrates the architecture of embedded system which consists of mother board, slave board, buses and some peripherals. The peripherals consist of touch screen, *universal serial bus* (USB), audio chip, flash memory, *synchronous dynamic random access memory* (SDRAM), *central processing unit* (CPU) and power devices. The capacity of storage in mother board includes 2MByte flash memory and 16MByte SDRAM. In order to communicate to other peripherals, the mother board can connect to slave board such as creator 920T-S3C2410, ARM, XScale PXA270, *etc.* The peripherals in slave board comprises of touch screen, USB and audio chip. If an application designs for using touch screen and audio, the mother board via bus communicates to slave board to controlling them. Figure 7 illustrates the architecture of master bus communicating to slave bus for embedded system. Figure 8 shows the experimental platform [10] that includes mother board, slave board and touch screen device.

The feasibility of the proposed middleware software was tested by two embedded systems, MP3 player system and e-book system. The specification of embedded system 1 comprises of CPU namely creator 920T-S3C2410, operating

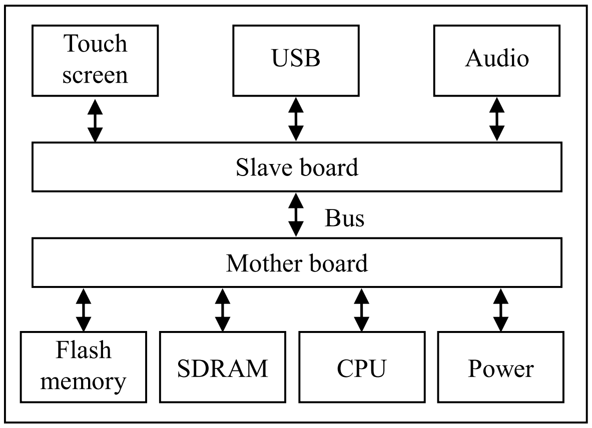

Figure 6. Architecture of embedded system.

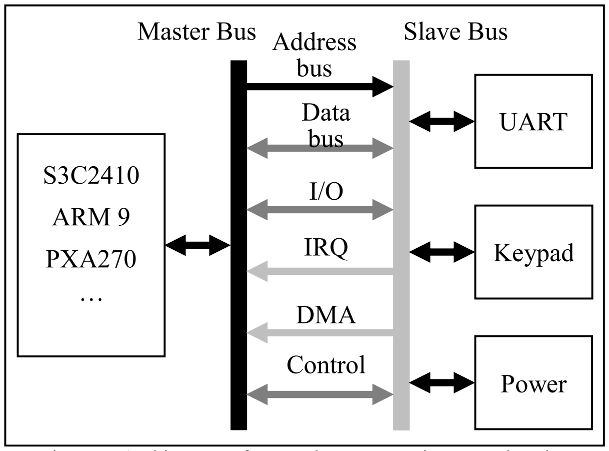

Figure 7. Architecture of master bus communicates to slave bus.

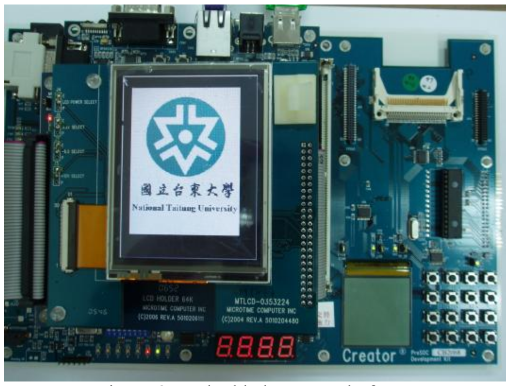

Figure 8. Embedded system platform.

system called embedded linux system, file system as EXT3, embedded *graphical user interface* (GUI) namely miniGUI [11] and middleware software. The other system name embedded system 2 that designs for e-book system. It distinguishes CPU as XScale PXA270 from embedded system 1. Others specifications such as operating system, file system, embedded GUI, *etc.* are similar to embedded system 1. The detail information of both embedded systems is exhibited in TABLE I.

Figure 9 shows the content information format for e-book system. Label 1 is a tag namely *book* which indicates an electronic book. Label 2 illustrates a tag as *bgm* that guide e-book system to play a music from the specify direction. Label 3 demonstrates a tag called *img* that will be parsed to load a picture. Label 4 shows a tag *p* that denotes e-book system to display a paragraph text. Therefore, label 5 exhibits a set of text. Label 6 displays an end tag related to label 4. Label 7 and 8 is same to label 3. Label 9 is an end tag related to label 1.

TABLE I. TESTBENCH OF EMBEDDED SYSTEMS

| **Specification** | **Embedded system 1** | **Embedded system 2** |
|---|---|---|
| CPU | S3C2410 | XScale PXA270 |
| Operating system | Embedded linux | Embedded linux |
| File system | Ext3 | Ext3 |
| Embedded GUI | MiniGUI | MiniGUI |
| Deploy middleware software | Yes | Yes |
| Applications | MP3 player system | e-book system |

```
1 <book>
2 <bgm src="img/3pig/3pig.mp3" />
3 <img src="img/3pig/pig1.jpg" />
4 <p>
5 Once upon a time there was a mother pig
who had three little pigs.The three little pigs
grew so big that their mother 07said to…
6 </p>
7 <img src="img/3pig/pig10.jpg" />
8 <img src="img/3pig/pig26.jpg" />
9 </book>
```

Figure 9. Content information format for e-book system.

## V. CONCLUSIONS

This paper presents the architecture of embedded system with middleware software. It also demonstrates middleware software for embedded system that consists of API module, service manager module and content manager module. In order to verify the feasibility of the proposed, two embedded system namely MP3 player system and e-book system are used to test the middleware software which deploys on embedded system. Experimental result shows the proposed is feasible on embedded system.

## ACKNOWLEDGMENT

The authors would like to thank the National Science Council of the Republic of China, Taiwan, for financially supporting this research under Contract No. NSC100-2221-E-143-006.

## REFERENCES

[1] Fernando Rincón, Jesús Barba, Francisco Moya, Félix J. Villanueva, David Villa, Julio Dondo, and Juan Carlos López, "System-Level Middleware for Embedded Hardware and Software Communication," in *Proc. of the Intelligent Solutions in Embedded Systems*, pp. 127-138, Jun. 21-22, 2007.

[2] W. Ces rio, L. Gauthier, D. Lyonnard, G. Nicolescu, and A. A. Jerraya, "Object-based Hardware/Software Component Interconnection Model for Interface Design in System-on-a-Chip Circuits," *Journal of Systems and Software*, vol. 70, no 3, pp. 229-244, Mar. 2004.

[3] Ronald Hecht, Stepah Kubish, Harald Michelsen, Elmar Zeeb, and Dirk Timmermann, "A Distributed Object System Approach for Dynamic Reconfiguration," in *Proc. of the Parallel and Distributed Processing Symposium* (IPDPS), Apr. 25-29, 2006.

[4] A. Gerstlauer, Communication Abstractions for System-Level Design and Synthesis, Technical Report CECS-TR-03-30, UC Irvine, 2003.

[5] Schulz-Key, C., Winterholer, M., Schweizer, T., Kuhn, T. and Rosenstiel, W, "Object-Oriented Modeling and Synthesis of SystemC Specifications," in *Proc of the Design Automation Conf.* (ASP-DAC), pp. 238-243, Jan. 27-30, 2004.

[6] Andreas Wolff, Stefan Michaelis, Jens Schmutzler and Christian Wietfeld, "Network-centric Middleware for Service Oriented Architectures across Heterogeneous Embedded Systems," in *Proc. of the IEEE EDOC Conf.* (EDOC 2007), pp. 105-108, Oct. 15-16, 2007.

[7] Wolff A and Michaelis S, "MORE System and Architecture", Public Deliverable, EU-Project MORE, Dortmund 2007.

[8] More project, http://www.ist-more.org.

[9] Irmert, F., Fischer, T., and Meyer-Wegener, K., "Runtime Adaptation in a Service-Oriented Component Model,". in *Proc of the 2008 international Workshop on Software Engineering For Adaptive and Self-Managing Systems* (SEAMS '08), pp. 97-104, May 12-13, 2008.

[10] Embedded system platform, http://www.microtime.com.tw/product/product.htm.

[11] Licia Capra, Wolfgang Emmerich and Cecilia Mascolo,"Middleware for Mobile Computing," UCL Research Note RN/30/01, July 2001.

[12] M. Caporuscio, Design, Development and Analysis of Distributed Event-Based Systems, The PhD thesis, Dept. of Computer Science, University of L'Aquila, Jan. 2007.

[13] A. Zarras and V. Issarny, “A Framework for Systematic Synthesis of Transactional Middleware,” in *Proc. of the International Conf. on Distributed Systems Platforms and Open Distributed Processing*, pp. 257-272, Sep. 01. 1998.

[14] W. Emmerich, Engineering Distributed Objects, JohnWiley & Sons, 2000.

[15] C. Szyperski, Component Software-Beyond Object-Oriented Programming, Addison-Wesley, 2002.

[16] Qilin Li, Wei Zhen, Minyi Wang, Mingtian Zhou and Jun He, “Researches on Key Issues of Mobile Middleware Technology,” in *Proc. of the International Conf. on Embedded Software and System Symposia* (ICESS), pp. 333-338, Jul. 29-31, 2008.

[17] Embedded graphical user interface system, http://www.minigui.org.