Survey of Open Source Operating System for the IoT Devises

Jalpa Patel
Assistant Professor
Department of Computer Engineering
Sal Institute of Technology and Engineering Research, Gujarat, India

Abstract— The Internet of Things (IoT) is the network of physical objects devices, vehicles, buildings and other items embedded with electronics, software, sensors, and network connectivity that enable these objects to collect and exchange data. IoT is the interconnection of Wireless Sensor Network (WSN) Devices with Internet space. They are generally small and battery-operated with memory of the order of 100 kilobytes. IoT devices are suitable with 8-bit microcontrollers which are far left behind by the present generation Windows/Unix/Mac-based desktops and laptops. This paper presents various types of operating system of IOT devises.

Key words: IOT, OS

I. INTRODUCTION

The demands of an increasingly data driven world mean that IOT device will require robust and reliable software. For that real-time operating system (RTOS) is needed. IOT devices as embedded systems transmit and receive information over a network while many embedded systems manage well with less sophisticated software and networked devices require more capable systems. The software for IOT device must be: *Scalable* to accommodate a wide range of different classes of devices, *Modular* to choose only the components to meet tight RAM requirements, *Connected* so data can be in and out of the device via Wi-Fi, Ethernet, USB, or Bluetooth and *Reliable* so device can be certified for safety-critical applications.

II. DESIGN ASPECTS OF OS IN THE IOT

In order to successfully adapt to the constraints of typical IoT devices, an OS must be designed purposely in all its aspects. In this section, we will thus analyze the key design aspects of OS in the IoT context and decompose the existing OS according to this analysis, in order to compare them.

There are multiple characteristics to categorize different types of OS. One of the most fundamental characteristic is the structure of the kernel. The OS can (i) be built in a monolithic way, (ii) follow a layered approach, or (iii) implement the microkernel architecture [8]. While a monolithic kernel is the simplest way to design an OS, it lacks modularity and often results in a complex structure that is hard to understand when the system exceeds a certain size. The layered model helps to better segment the system in a hierarchical way. The developer has to choose the level of separation between kernel and user space. In the microkernel, design goes even further in modularity: the whole OS is split up in small, well-defined modules, and only a minimum set of functions runs in kernel mode. This approach increases the reliability of the system as bugs in individual components (such as device drivers or the file system) will not crash the system.

III. EXISTING OPEN SOURCE OPERATING SYSTEM

This section list the existing open source operating system.

A. Contiki [3]

It is an open source operating system for networked. It is a memory-constrained system with a particular focus on low-power wireless IoT devices. Examples of where Contiki is used include street lighting systems, sound monitoring for smart cities, radiation monitoring systems, and alarm systems.

B. RIOT [8]

Is an open-source microkernel operating system for the IoT. RIOT is an operating system based on microkernel architecture. RIOT's kernel is mostly inherited from Fire Kernel, a kernel which was originally developed for sensor networks.

C. Tiny OS [4]

TinyOS is a free and open source software component-based operating system and platform targeting wireless sensor networks (WSNs). TinyOS is an embedded operating system written in the nesC programming language as a set of cooperating tasks and processes. It is intended to be incorporated into smart dust.

D. Lite OS [7]

LiteOS is a real-time operating system (RTOS). LiteOS is a Unix-like operating system that fits on memory-constrained sensor nodes. This operating system allows users to operate wireless sensor networks like operating Unix, which is easier for people with adequate Unix background. LiteOS provides a familiar programming environment based on Unix, threads, and C. It follows a hybrid programming model that allows both event-driven and thread-driven programming.

E. Mantis OS [6]

Mantis OS is implemented in C. It has services resembling to a subset of POSIX implementation. Threads are assigned certain priority levels. Within a priority level, Round Robin (RR) scheduling is used.

F. Nano-RK [5]

In C-based Nano-RK, tasks follow fixed-priority scheduling. There are two types of priority scheduling. In the case of rate monotonic scheduling, tasks are assigned priorities statically depending on the period of the job. Shorter jobs have higher priorities. While rate harmonized scheduling is aimed at saving power, tasks are grouped together for execution to eliminate any idle cycles. Dynamic memory management and MPU are absent. The OS supports a lightweight protocol stack.

IV. COMPARISON OF IOT OS

OS	Dev. Language	Scheduling	Real Time System
Contiki	Subset of c	preemptive	Partial
RIOT	C, C++	Priority Base Tickless	Yes
TinyOS	Nes c	Preemptive FIFO, EDF	No
LiteOS	C, Lite C++	Priority Based, RR	Partial
Mantis OS	C	Priority Based, RR	Partial
Nano-RK	c	Preemptive Fixed Priority Based	Yes

Table 1: Comparison of Operating System of IOT devices

V. CONCLUSION

In this survey, different types of OS for IOT devices have been observed and Design aspects of OS in IOT are also listed. In this paper I have focus on Open source IOT Operating System.

REFERENCE

[1] Hahm, Oliver, Emmanuel Baccelli, Hauke Petersen, and Nicolas Tsiftes. "Operating Systems for Low-End Devices in the Internet of Things: a Survey." (2015).
[2] Baccelli, Emmanuel, Oliver Hahm, Mesut Günes, Matthias Wählisch, and Thomas C. Schmidt. "RIOT OS: Towards an OS for the Internet of Things." In Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on, pp. 79-80. IEEE, 2013.
[3] Dunkels, Adam, Björn Grönvall, and Thiemo Voigt. "Contiki-a lightweight and flexible operating system for tiny networked sensors." In Local Computer Networks, 2004. 29th Annual IEEE International Conference on, pp. 455-462. IEEE, 2004.
[4] Levis, Philip, Sam Madden, Joseph Polastre, Robert Szewczyk, Kamin Whitehouse, Alec Woo, David Gay et al. "TinyOS**:** An operating system for sensor networks." In Ambient intelligence, pp. 115-148. Springer Berlin Heidelberg, 2005.
[5] Eswaran, Anand, Anthony Rowe, and Raj Rajkumar. "Nano-rk: an energy-aware resource-centric rtos for sensor networks." In Real-Time Systems Symposium, 2005. RTSS 2005. 26th IEEE International, pp. 10-pp. IEEE, 2005.
[6] Bhatti, Shah, James Carlson, Hui Dai, Jing Deng, Jeff Rose, Anmol Sheth, Brian Shucker, Charles Gruenwald, Adam Torgerson, and Richard Han. "MANTIS OS: An embedded multithreaded operating system for wireless micro sensor platforms." Mobile Networks and Applications 10, no. 4 (2005): 563-579.
[7] Cao, Qing, and Tarek Abdelzaher. "LiteOS: a lightweight operating system for C++ software development in sensor networks." In Proceedings of the 4th international conference on Embedded networked sensor systems, pp. 361-362. ACM, 2006.
[8] Baccelli, Emmanuel, Oliver Hahm, Mesut Günes, Matthias Wählisch, and Thomas C. Schmidt. "OS for the IoT-Goals, Challenges, and Solutions." In Workshop Interdisciplinaire sur la Sécurité Globale (WISG2013). 2013.

