Wireless Smart Sensor Nodes using Embedded Linux Kernel on ARM Platform

Sibi. P. N
VLSI and Embedded Systems,
Mangalam College of Engineering,
Ettumanoor, Kerala

Anjali V
Asst. Professor, EC Dept.,
Mangalam College of Engineering,
Ettumanoor, Kerala

Vishnu S
Senior Engineer, SEG
CDAC,
Thiruvananthapuram, Kerala

Abstract— **Recently, advancements of the wireless communication and microelectronic technologies enable the interconnection between devices and automatic operations. Additionally, sensing physical information based on the wireless sensor device helps more intelligent services to incorporate the ubiquitous paradigm. The aim of this paper is to develop a new smart device with the sensing ability as well as the wireless communication.**

Keywords— Wi-Fi; Linux; sensor; kernel; ARM; Cortex-A8

I. INTRODUCTION

Day by day, the technology is becoming an increasingly important part of our everyday life. With the advancements of MEMS (Micro Electro Mechanical Systems) technology, the ubiquitous paradigm [1] can be incorporated to home and office networks and it is becoming the attractive research issues in the industry areas. To provide more intelligent services, these networks need the new smart devices which have processing and wireless networking abilities. Intelligent wireless sensor-based controls have drawn industry attention on account of reduced costs, better power management, ease in maintenance, and effortless deployment in remote and hard-to-reach areas [8]. They have been successfully deployed in many industrial applications such as Maintenance, monitoring, control, security, etc [2].

The design is composed of two main components - a sensor-wireless hardware interface and system integration framework, which eases the definition of interaction between sensors based on process needs. The intelligence necessary to assert away the detector signals, monitor the functions against defined operational templates, and enable the swapping of sensor and RF link resides on the ARM of the hardware interface. The system is based on the Internet with hardware implementations which is proposed. It is also a low-cost solution and has a scalable structure which allows new appliances to be added with no major changes to its core.

In recent years, with the development of information technology, the embedded system has been widely applied to various aspects of social life, such as mobile computing equipment, network equipment, household appliance, instruments, etc. In many embedded operating systems, Linux, with its unique advantages, gets hold of a heavy parcel of the embedded field, and has become the world's second-biggest operating system. Linux has open source and rich software resources, supports multi-thread, multi-process, multi-user, and has good portability, stability and powerful functions. It supports a large number of microprocessor architecture, hardware devices, graphics support and communication protocols, etc. [4]. Using Linux as a bottom operating system and taking modification on OS, can satisfy different requirements of specific applications. Linux delivers a big potential in the embedded arena, which is loved by many businesses, scientific research units [9].

The system based on this paper uses SimpleLink CC3000 Wi-Fi Module and AM3359 Cortex- A8 ARM processor which controls the wireless transceiver. The Wi-Fi module uses infrastructure mode network of IEEE802.11g. The common constraints that we are facing during the design and implementation of Wi-Fi networks are security issues, low data rates, distance problem, error rate and hidden node problems. These negative aspects can be simplified by using SimpleLink CC3000 Wi-Fi Module.

The Cortex-A8 high-performance processor is proven in end devices today. From high-end feature phones to netbooks, DTVs, printers and automotive-infotainment, the Cortex-A8 processor offers a proven high-performance solution with millions of units shipped annually [5]. The ARM Cortex™-A8 processor, based on the ARMv7 architecture, has the ability to scale in speed from 600MHz to greater than 1GHz. The Cortex-A8 processor can meet the requirements for power-optimized mobile devices needing operation in less than 300mW; and performance-optimized consumer applications requiring 2000 Dhrystone MIPS.

The aim of this paper is to develop a new smart device with the sensing ability as well as the wireless communication.

II. MATERIAL AND METHODOLOGY

A. *Linux System Architecture*

The Linux operating system is composed of four major subsystems: User Applications, O/S Services, Linux Kernel and Hardware Controllers. Despite its large code base (over seven million lines of code), the Linux kernel is the most flexible operating system that has ever been created [10]. It can be tuned for a wide range of different systems. By customizing the kernel for your specific environment, it is possible to create something that is both smaller and faster than the kernel provided by most Linux distributions.

In this paper, the methods and progress of transplanting the embedded Linux operating system to the ARM hardware platform are described. It includes the establishment of cross-compiler environment, the reduction and compilation of startup code as well as Linux kernel, and the construction of the root file system. The key point focuses on the structure and function of boot loader as well as the transplantation of Linux kernel.

B. The Arm Platform

The simplicity of ARM processors makes them suitable for low power applications. As a result, they have become dominant in the mobile and embedded electronics market, as relatively low-cost, small microprocessors and microcontrollers. The ARM Cortex-A8 processor is a high-performance, low-power, cached application processor that implements the ARMv7- A architecture profile and provides full virtual memory capabilities [6].

C. Sensors

Redundant and complementary sensor data can be fused and integrated using multisensory fusion techniques to enhance system capability and reliability in the case of sensor error or failure. Smart sensors are basic sensing elements with embedded intelligence (combination of a sensing element with processing capabilities offered by a microprocessor), which can perform one or more of the following functions such as Logic functions, Two-way communication or Make decisions. The accurate analog-output temperature sensor which is used for this particular application is DS600.

D. Device Driver Development

Linux has helped to democratize operating systems. Device driver is an independent module, which can shield the details of physical device work completely, and provide a common programming interface. A device driver has three sides: one side communicates to the rest of the kernel, one communicates to the hardware, and one talks to the user as shown in figure 1 [7]. Building a hardware device driver and successfully integrating it into a system requires the following steps: (1) Learn and understand the hardware, (2) Design the driver, (3) Code the driver, (4) Debug, (5) Integrate and (6) Document the Functional and interface description, Restrictions, examples of use, source code, compilation, linking, and installation process.

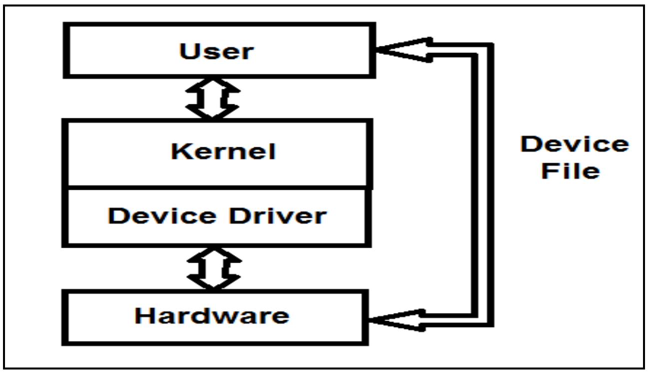

Fig. 1: Anatomy of a Device Driver

E. Wi-Fi Based Wireless Transmission

Choosing a better technology might depends on what you want to do. Wi-Fi wireless LAN adapters are much more powerful compared to other technologies; data transmission rates reaching about 54Mbps. Therefore, this paper proposes a system which employs a very low power Wi-Fi device. By using it, the user can control the light, the temperature, can detect the dust air density and the humidity and pressure can be monitored. The current system based on this paper uses TI CC3000 module for the wireless connectivity and it is a self-contained Wi-Fi solution that enables internet connectivity for a wide variety of microcontroller (MCU) systems.

The flow chart of the proposed Linux software development is shown in figure 2.

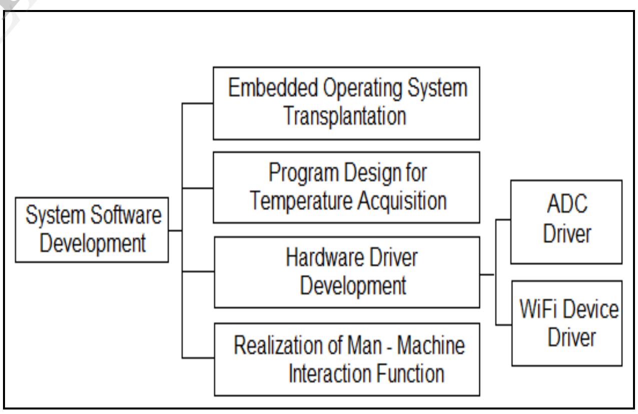

Fig. 2: Software development flow chart

The temperature is obtained using the temperature sensor and it is given to the ADC which is built-in with AM 3359 and the corresponding digital value is given to the remote PC using a Wi-Fi connection.

III. IMPLEMENTATION AND RESULTS

It is possible to develop and build the applications directly on the target board in which the ARM processor lies, but it is certainly not an efficient use of time. The processor will eventually begin to act as a limit on how quickly we can compile. To fix this problem, we would like to be able to develop and build our projects on another machine and then easily transfer to the target board. BeagleBone Black is used as the target board since the application requires a low cost ARM Cortex-A8 based processor.

A. Toolchain Installation

If we evolve from a Linux platform, it should be capable to work on any Embedded Linux system; therefore Installed a Linux distribution (distro) such as Ubuntu on the Non-Linux System, since it is the most popular distro [11]. To get our processors speaking the same language we need to cross-compile for the target board by using a compiler that is targeted at the ARM architecture. For this, downloaded the arm-eabi-toolchain and installed the dependencies and the library libexpat; without this library, the toolchain will compile successfully, only the debugger won't be configured to load software to the correct location in the board's memory [12].

B. ADC Driver Development

The Analog to Digital Converter (ADC) on the BeagleBone is both a Touch Screen Controller (TSC) and a general purpose ADC. After enabling the driver we can read the analog inputs on the target board. The analog inputs which are connected to the temperature sensor outputs should now be activated and exported to the file system. Cat the files to read their values.

C. Wi-Fi Driver Development

An RF cape is used to fit the CC3000 module to the target board's GPIO connectors [3]. After downloading the Smart Config Application for the Android, which is used to associate CC3000 to an access point which is already connected to an Android device. Connect the phone to the access and start the Smart Config application and make sure the correct information is filled in. Once a connection has been made a static page which includes the corresponding digital value of the temperature sensor output is transmitted and the socket is then closed. The program then loops back and waits for the next connection.

In a similar way redundant and complementary sensor data can be fused and integrated using multi smart sensor nodes to enhance system capability and reliability in the case of any sensor error or failure.

IV. CONCLUSION

A new smart device with the sensing ability as well as the wireless communication is developed. The Linux drivers for ADC and CC3000 Wi-Fi module have been developed and the kernel is modified subsequently.

ACKNOWLEDGMENT

Authors are highly indebted to CDAC, Thiruvananthapuram and express special gratitude and thanks to industry persons for furnishing necessary data regarding this paper work.

REFERENCES

[1] M. Weiser, "The Computer for the 21th Century," in IEEE Pervasive Computing Magazine, Jan. - Mar. 2002.

[2] Madni, AM., "Smart configurable wireless sensors and actuators for industrial monitoring and control," Communications, Control and Signal Processing, 2008. ISCCSP 2008. 3rd International Symposium on , vol., no., pp.447,448, 12-14 March 2008.

[3] http://processors.wiki.ti.com/index.php/CC3000_Module_with_AM335x_Using_Linux.

[4] Chen lijun, "Understanding Linux kernel source code deeply"[M], Beijing: Posts & Telecom Press. 2002.

[5] http://www.arm.com/products/processors/cortex-a/cortex-a8.php

[6] http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.set.cortexa/index.html

[7] http://www2.diees.unict.it/users/bando/files/smart_sensors.pdf

[8] H. Rammamurthy, S.B. Prabhu, R. Gadh , and A. M. Madni, "Wireless Industrial Monitoring and Control Using a Smart Sensor Platform," IEEE Sensors Journal, vol. 7 No.5, pp. 611-618, May 2007.

[9] http://www.scientific.net/AMR.542-543.1023

[10] http://www.linuxtopia.org/online_books/linux_kernel/kernel_configuration/ch02.html

[11] http://www.michaelhleonard.com/cross-compile-for-beaglebone-black/

[12] http://www.tincantools.com/wiki/BeagleBone_Black_Eclipse_and_GDB