International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 1, January - 2014

Review on GPIO Device Driver Development on Embedded Linux Platform

Vidhi Z. Patel[1,] Bharat V. Tank[2]

[1]*M.E. Student, Parul Institute of Technology, E&C Dept*
[2] *Assistant Professor, Parul Institute of Technology, E&C Dept.*

Abstract

Embedded Linux is operating system, like other OS it contain kernel inside. so in this case we have to use microprocessor. Now days Arm cortex is very popular for Embedded Linux. In Linux, kernel provides resource for hardware and software. If we would like to use hardware resources than first we have to make device driver and built and resister with kernel or kernel subsystem with different type of interface like procfs, sysfs and device node itself with major number and minor number. With the help of module programming we can add or remove device driver dynamically, So with the help of GPIO device driver we can use GPIO as input or output as per our requirement, that way we will know that it's general device driver and also we will make Application code for reading and writing this device driver.

Keywords: Raspberry Pi Board, Ubantu 12.04, GPIO Driver

1. Introduction

Linux for embedded systems is about the use of Linux kernel-based operating systems on embedded systems such as customer-premises equipment, in-vehicle infotainment (IVI), networking equipment, machine control, industrial automation, navigation equipment and medical instruments in general.

Linux has been ported to a variety of CPUs which are not only primarily used as the processor of a desktop or server computer, but also ARM, AVR32, ETRAX CRIS, FR-V, H8300, IP7000, m68k,MIPS, mn10300, SuperH, and Xtensa processors, It is also used as an alternative to using a proprietary operating system and tool chain.

With the availability of consumer embedded devices, communities of users and developers were formed around these devices: Replacement or enhancements of the Linux distribution shipped on the device has often been made possible thanks to availability of the source code and to the communities surrounding the devices. Due to the high number of devices, standardized build systems have appeared like Open Embedded, Build root.

Embedded operating system is important to the operation of an embedded system, environment and development platform, whether it is efficient, stable, secure and so will have a direct bearing on the success or failure of embedded systems has become an embedded system design and development priorities. But Linux developed for desktop machines, which is used in embedded systems have some differences, such as memory capacity and limited compared to desktop computers, so how to transform Linux into a small capacity, high stability and easy the development of embedded operating system becomes a critical issue. This also means that the embedded Linux operating system, used in digital products and industrial control fields of a lot of work needs to be done.

2. Previous Approach

Device driver is very important in embedded system to operate or control a particular device that is attached to the computer. All peripheral devices have different device driver. The work presented in[1] introduced the Linux 2.6.11.6 built on Panda Board which is ARM based development board in which porting of Kernel on processor done successfully. Work presented in [2] making one device driver to control an LED according to our application program. On LED Matrix hardware Led character device driver is implemented using Linux operating System on ARM 9. With the support of device driver Led Matrix device work properly. Device driver actually controls the I/O devices. In [3], Embedded Linux operating system is implemented on ARM-11 for measuring the real time parameters like temperature, humidity and light. In this they generate the Sensor driver for Light, Humidity and Temperature. The work presented in [4] For Video acquisition technology, CMOS Camera driver is implemented on ARM 9 processor on platform of Linux 2.6.32.For data acquisition applications, USB device driver is also successfully implemented on ARM 9 processor on platform Linux 2.6.32.

3. Objectives

The main objective of this review paper is firstly to built the Linux on Raspberry Pi board and secondly to develop a GPIO Device Driver for character devices which are attached to the Raspberry Pi board on GPIO pin port.

4. Proposed approach

In Modern technology, people like to use multitasking and real time processes in their applications. So, with the help of Embedded system and Linux operating system we can get the real time operating system and multitask also improve speed of the system which is very important in this fast life style.

First to build an embedded Linux Kernel in Microprocessor ARM CORTAX Series which is already SoC in Raspberry Pi Board support package for building the Linux Kernel on Raspberry Pi board we need source code which is obtained from Official Website of Kernel. Embedded Linux development broadly involves three tiers: the boot loader, the Linux kernel, and the graphical user interface (or GUI).

For building the Linux kernel, cross-compiler should be accessible on your execution path. Up to this step, Device has a limited functionality, as it just enables the kernel to boot and send debug messages through the configured serial port[1].

For a full fledged embedded module it is quit elementary without the support for various peripherals through device drivers which are paramount to the module. Linux kernel though monolithic has an excellent modular approach which enables the driver modules to be attached and detached at run time itself. For this we have to do module programming for device driver using C, C++ or Java Script.

Basic block diagram of Linux system is shown in Fig 1 which Linux is basically divided in two region of memory.

- User Space
- Kernel Space

The kernel provides certain services which are strictly reserved for running privileged kernel, kernel extensions, and most device drivers, and *user space* that is, everything outside the kernel, both libraries and application programs, uses these. Programs in user space contain system calls that ask the kernel to do something, and the kernel does so, or returns an error code. Application programs do not usually contain direct system calls. Instead, they use library calls and the library uses system calls. But an application program can construct a system call "by hand".

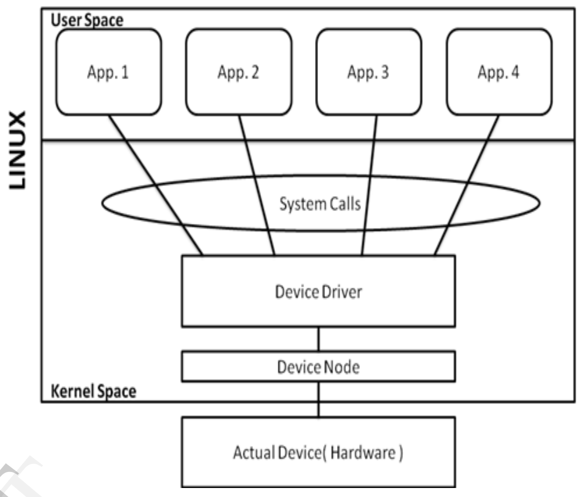

Fig 1: Basic Block Diagram of Linux Architecture and Embedded System

A device driver is a computer program that operates or controls a particular type of device that is attached to a computer. Drivers are hardware-dependent and operating-system-specific. They usually provide the interrupt handling request for any necessary time-dependent hardware interface. When a calling program invokes a routine in the driver, the driver issues commands to the device.Once the device sends data back to the driver, the driver may invoke routines in the original calling program. Instead of putting code to manage the HW controller into every application, the code is kept in the Linux kernel. It abstracts all Hardware device regular files in the kernel for handling of devices. Device node allow software to interact with a device driver using standard input/output system calls, which simplifies many tasks and unifies user-space I/O mechanisms. Device node often provides simple interfaces to peripheral devices, such as printers and serial ports. But they can also be used to access specific resources on those devices, such as disk partitions. Finally, device files are useful for accessing system resources that have no connection with any actual device such as data sinks and random number generators.

5. Applications

- Monitoring MMC/SD card insertion/removal
- Detecting card write protect status
- Configuring a transceiver
- Bit banging a serial bus
- Poking a hardware watchdog
- Sensing a switch, and many more

6. Conclusion

The purpose of this review paper is to investigate the general purpose driver in Linux operating system. For different devices different device driver will used. After reviewing research paper it is possible to develop a general device driver for GPIO Controller to Plug and Play devices that can be dynamically connected to and disconnected from a hardware platform, a GPIO controller device is permanently attached. In addition, connections between GPIO pins and a peripheral device are assumed to be permanent.

7. References

[1] Pratyusha.Gandham and Ramesh N.V.K "Porting The Linux Kernel To An Arm Based Development Board"International Journal of Engineering Research and Applications (IJERA May- June 2012)
[2] G.Sravani, B. Karunaiah and Prof K V Murali Mohan "Implementation Of Led Driver For Commercial Applications Based On Arm9" International Journal of Engineering Research and Applications (IJERA Nov-Dec 2012)
[3] Hong Shao, Chen Yang, Na Zong, Ke-feng Jin "Design of a Hospital Environment Data Acquisition System Based on ARM11 and Embedded Linux" International Conference on Computer Science and Electronics Engineering (ICCSEE 2013)
[4] CH. P. N. S. Sujitha, DVSR Sesidhar "Developing CMOS Camera and USB Device Drivers in Linux 2.6.32" International Journal of Electronics Communication and Computer Technology (IJECCT-2013)