Journal of Theoretical and Applied Information Technology
30th June 2013. Vol. 52 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

KERNEL LEVEL HARDWARE BASED SINGLE CHIP COMPUTER AND DEVICE CONTROL

[1]DR.M.P.CHITRA, M.E, PHD, [2]DR.N.R.SHANKER, B.E, M.TECH, PHD

[3]B.PREETHI, [3] M.SANU MURUGAN, [3] M.SOUNDARYA

[1]HOD, Department of Electronics and Communication Engineering, Panimalar Institute of Technology

[2]R&D Engineer, Chase Technologies

[3]Project members, Department of Electronics and Communication Engineering, Panimalar Institute of Technology

E-mail: [1] chi_mp2003@yahoo.co.in, [3] prettybask@gmail.com,

ABSTRACT

In this paper the implementation of a kernel level hardware based single chip computer and device control has been proposed. The dc motor control based on the concept of PWM using the 8bit microcontrollers-Arduino has been analysed. The Arduino have no operating system and the toolkits such as Arduino IDE, Eclipse is used. The programming language that the Arduino supports is C++. The speed of Arduino is low about 16MHz and the RAM space is only 2 Kbytes. The Arduino have no USB ports. The drawbacks of the 8bit microcontroller are overcome by using the 32bit microcontroller, Raspberry Pi which uses the Linux kernel-based operating systems. In the Raspberry Pi the toolkits such as Scratch box, QEMU, Open Embedded is used. The programming language used in Raspberry Pi is Python language. Unlike Arduino the programming language in Raspberry Pi is not restricted only to C or C++, any language which can be compiled by ARMv6 can be used. The AM space is about 256 MB and it has two USB ports. The Raspberry Pi has audio input via USB.The speed of Raspberry Pi is increased to about 700 MHz

Keywords — *Arduino, Broadcom BCM2835, Debian, Linux, Raspbian Distro, RTAI, RTLinux-THIN*

1. INTRODUCTION

An embedded system is a computer system within an immense system and it is designed for precise control functions, frequently with real-time computing limits. It is entrenched as fraction of an absolute device together with hardware components. By disparity, a general-purpose computer is designed to be supple and it meets the large range of end-user needs. Processing cores such as digital signal processors are recognized in embedded systems. The significant characteristic of embedded systems is committed to handle a exact task. The advantages of the embedded system are that the size, price of the product is reduced and the reliability, performance is increased. Some embedded systems benefits the economies of scale. Bodily, embedded systems is vast, ranging from movable devices such as digital watches, MP3 players, to huge immobile installations like traffic lights, factory controllers. The difficulty varies from a distinct microcontroller chip to multiple units and networks which are mounted within a large framework or enclosure. For the purpose such as safety and usability real-time performance constraints should be met by user. In case there are no performance requirements, the system hardware is simplified in order to reduce the costs. In several embedded systems the small, automated parts is entrenched within a larger device hence embedded systems are not constantly separate devices. Most embedded systems comprises of small, mechanized parts inside a larger device that serves as an universal purpose. Embedded Linux is the exploitation of Linux in embedded computer systems. Some of the examples are mobile phones, media players, set-top boxes and other devices, networking equipment, machine control, industrial automation, navigation instruments and medical equipments. The Linux has been transferred to a range of CPUs which are used as the processor of a desktop and also as ARM, AVR32, SuperH and Xtensa processors. Instead of using patent operating system and tool chain the Linux can be used. The Embedded systems ranges from no user interface to

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

intricate graphical user interfaces that bear a similarity to modern computer desktop operating systems. The no user interface performs only one task. The simple devices such as buttons, LEDs, graphic LCDs like popular HD44780 LCD are being used in simple embedded system. Linux is the widely used operating system as it has superior performance, flexibility, speed and lower costs. In November 2008 Linux held an 87.8 percent share of the world's top 500 supercomputers refers to the specialized Linux distributions for desktop personal computer users is referred to as Desktop Linux which includes a graphical user interface and personal use applications. The target for the Linux distributions varies from the specific desktop role and all the software available for the platform. When the operating system is installed the user may select either a "desktop" or "server" type.

Figure 1: Ubuntu, a popular Linux distribution

Program such as Synaptic or Package Kit is provided by Linux for browsing free software applications and it is already tested and configured for a specific distribution., Ranging from embedded systems to supercomputers [1]. Linux systems are being used in every area and have obtained a position in server installation frequently using the accepted LAMP application stack [2] . Utilization of Linux distributions in residence and venture desktops has been rising.[3]Linux distributions in the Net book with countless devices such as the ASUS Eee PC and Acer Aspire have become very popular. Installation of Acer Aspire one shipping with modified Linux distributions have been performed [4]. Most of the Linux distributions support variety of programming languages. The progressing utensils used for building Linux applications and operating system programs are found inside the GNU tool chain. The GNU tool chain the GNU Compiler Collection (GCC) and the GNU build system. While using minimizing space, the flexibility is provided by using a graphical touch sensing screen or screen-edge buttons hence the sense of the buttons can vary with the screen, and choice is ordinary actions of pointing on what is required. For a pointing device there exists a screen with a "joystick button" in the handheld systems. The user interface can be provided remotely by using a serial connection such as RS-232, USB, I²C,etc or network connection such as Ethernet connection. The advantages of this approach are that the potentiality of embedded system is extended, the cost of a display is avoided, BSP is simplified and renders better user interface on the PC. The grouping of an embedded web server operating on an embedded device for instance an IP camera otherwise network routers is the best example. Therefore there is no need of installing bespoke software as the user interface is displayed in a web browser on a personal computer coupled to the device. Arduino is a 8 bit microcontroller and it is designed for using electronics in various projects. The hardware of the Arduino comprises of an open basis hardware board and it is designed with 8-bit Atmel AVR microcontroller. The software comprises of a benchmark programming language compiler with a boot loader. The Arduino project is a division of the open source wiring platform [5]. The Arduino is programmed by means of a Wiring-based language, like C++ with simplifications and modifications. In the year 2008, the Arduino Duemilanove was announced [6]. In March 2009, the Arduino Mega was announced [6].As of May 2011, above 300,000 Arduino units was in operation all over the world [7]. An Atmel 8-bit AVR microcontroller with complementary components is present in the Arduino board to assist programming and merging into other circuits. In the Arduino the connectors are exposed in the standard way so that the CPU board can be linked to different compatible add-on modules. The modules are known as shields. Some shields respond directly with the Arduino board over various pins, but many are in isolation and addressable by means of an I²C serial bus hence shields can be arranged and used in parallel. The mega AVR sequence of chips, specially the ATmega8, ATmega168, ATmega328, ATmega1280, and ATmega2560 have been used by the Official Arduino. The Atmel ATmega644 microcontroller was published in 2006. The early concept of Raspberry Pi was based on the Atmel

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ATmega644 microcontroller. The Raspberry Pi acts as a computer when plugged into a Television and a keyboard and it is credit card sized. The Raspberry Pi uses Linux kernel-based operating system. The PCB layouts are offered for public download. The size of the first ARM prototype version of the computer is same as a USB memory stick. It has a USB port on one end and the High-Definition Multimedia Interface port on the other. Model A has only one USB port and no Ethernet controller, but it can be connected to a network with the help of a user-supplied USB Ethernet or Wi-Fi adapter. Model B has two USB ports and a Ethernet controller. There is no big variation between a model A and model B. If model A is with an external Ethernet adapter it would act as Model B. The general USB keyboards and mouse are compatible with the Raspberry Pi. In Raspberry Pi for file time and date stamping an operating system must apply a network time server, since Raspberry Pi does not accompany a real-time clock. Though, a real-time clock with battery can be added via the I2c interface. The Model B has a MIPI camera interface (CSI). The image was based upon Debian 6.0 Squeeze, with the Lightweight X11 Desktop Environment desktop and the Midori browser and various programming tools. The Raspberry Pi can be followed on different other platforms since the image also runs on QEMU. For hooking it up to a previous analogue Television there is a complex HDMI out on the board, to a digital Television using an inexpensive adapter for the DVI. The System on Chip is a Broadcom BCM2835. This contains an ARM1176JZFS, running at 700MHz, and a Video core 4 GPU. The GPU is competent of Blu-ray quality playback at 40MBits/s. Most devices will run at 800MHz. The over clocking options can be changed on first boot and any time later by running "raspi-config" in the recent Raspbian distro. These are experimental settings and every board won't be able to run firmly at the highest setting. By reducing the over clocking settings stability is restored. Broadcom which acts as the chip at the heart of the Raspberry Pi does not make public a complete datasheet for the BCM2835. A datasheet for the SoC is released which will cover the hardware on the Raspberry pi board e.g. the GPIOs. There is a standard 3.5mm jack optimised for audio out. Any supported USB microphone can be added for audio in. The Raspberry Pi is built from business chips which are capable for different temperature ranges. The LAN9512 circuit précised by the manufacturers is practised from 0°C to 70°C.

2. RELATED WORK

On improving real-time interrupt latencies of hybrid operating systems with two-level hardware interrupt [8]. The worst-case real-time interrupt latency for RTAI and the main factor for its optimization are recognized. The real-time kernel and the time sharing OS kernel is pooled and the methodology for implementing hybrid operating system is proposed. The significant issues for the completion are discussed based on the methodology proposed. Finally, by combining ARM Linux kernel 2.6.9 and µC/OS-II the implementation of a RT Linux-THIN on the ARM architecture is done. The scheme provides an simple method for implementing hybrid systems and achieves the performance enhancement for the time sharing and real-time subsystems.The time-sharing subsystem is regularly rooted in commodity operating systems, such as Linux [9], [10], [11] so that the development and maintenance costs are relatively low. The commodity operating systems since focus on general-purpose computing, hence it should be effectively utilised in real time environments such that the predictability of real-time applications is not impaired. In RTAI, the execution occurs only when there is no real time task to run that is the Linux OS kernel is treated as the inoperative task and is inactive. The purpose of the interrupt controller is emulated by modifying the interrupt-handling code hence the real time interrupt is never blocked by the Linux task or stop itself from being pre-empted [10].The use of interrupt-handling code for separating real-time and non-real-time interrupts and emulating the interrupt controller generates several problems. Interrupt disabling is often used in interrupt handlers in the Linux task. The interrupts are not really disabled and that the real time interrupt in RTAI is not blocked. However, based on the software emulation of the interrupt controller a flag is set in the interrupt-handling code and the interrupt disabling is done. So non-real-time interrupts of the Linux task can be delivered on the condition that they should be disabled during interrupt disabling. The interrupt can be turned off only individually even though interrupts will be turned off later, if the flag is set in the interrupt handling code. Because of these interrupt responses and processing the performance of the Linux task is degraded but the unpredictability of the real-time subsystem is increased. The size of the interrupt-handling code depends on the functions of identification and emulation. The interrupt handling code cannot be easily locked into cache if the code size is too big and various cache locking techniques

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[12], [13] may not be applicable as the predictability problem arises. The code in the Linux OS kernel must be rewritten even though a hardware abstraction layer (HAL) is preoccupied in RTAI [10]. The porting of a hybrid system to numerous processors becomes tedious. In order to overcome these problems hybrid operating systems based on two-level hardware interrupts is proposed which is easier rather than using hardware to separate real-time and non-real-time hardware interrupts by hardware. The main objective is to improve the predictability and real-time interrupt latency of the real-time subsystem, and to enhance the performance of the time-sharing subsystem. The Two-level hardware interrupts in high-end embedded processors have different interrupt request entries such as those based on the ARM architecture [14]. The performance is improved for both the time-sharing and real-time subsystems with the help of this architecture. When, the interrupt is disabled in Linux task the disabling/enabling bit is not set in the PSW instead in the virtual interrupt controller. If the interrupt disabled is non-real-time interrupts they are recorded into a First-In-First-Out queue, then it is acknowledged but not served. These non-real time interrupts will be served one by one depending on whether the disabling/enabling bit in the virtual interrupt controller is cleared. The processing flow of the interrupt distribution routine in RTAI is shown in Fig. 2.In the interrupt distribution routine, when an interrupt request is responded, the first step is to determine whether it is a real-time interrupt or non-real-time interrupt. In real-time interrupt, the corresponding interrupt service routine is called and served directly. In case of non-real-time interrupt the existence of running real-time tasks is noticed. If the non-real-time interrupt exist it is recorded into a FIFO queue, acknowledged but it is not served, as the real-time tasks have higher priority. Another method is to check the interrupt disabling/enabling bit in the virtual interrupt controller is set. The interrupt will be served and the corresponding interrupt routine will be called if the bit is not set; otherwise, the interrupt request is recorded and acknowledged but not served. Still the non-real-time interrupts can be responded when disabled during interrupt disabling which results in CPU overhead. The software method may result in some problems for systems with level-triggered interrupts. In a level triggered hardware interrupts, the existence of an un-serviced interrupt is indicated by the level of the interrupt request line. A device can signal an interrupt, by driving the line to the active level, and holding it in the same level until the interrupt is serviced. Therefore, the first step is to serve and then acknowledge driving the line to the inactive level but in software emulation method, acknowledgement is done and then served. So the level-triggered interrupts cannot be handled correctly. On this condition the interrupt handler cannot acknowledge that interrupt request directly until the request is serviced. Actually, if the acknowledgement is done directly by driving the line to the inactive level, it denotes that the interrupt request is already served. If the acknowledgement of the interrupt request is not done, until the interrupt request line is in the active level the interrupt will be triggered repeatedly hence the progression of the system is affected.

Figure 2:The processing flow of the interrupt distribution routine in RTAI

For the hybrid system the most important performance metrics is the real-time interrupt latency to provide predictable real-time services. The interrupt latency is defined as the time interval between the point where the assertion of the interrupt starts and the point where the execution of the corresponding interrupt service routine occurs. The Worst- Case Execution Time (WCET) is analysed for real-time interrupt latency as predictability is the major concern. Assume that, I1;

Journal of Theoretical and Applied Information Technology
30th June 2013. Vol. 52 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

I2; . . . ; IN are the N real time interrupts andI1 > I2 > _ _ _ > IN is the priority order of N interrupt, that implies that if I1; I2; . . . ; IN occur simultaneously, is I1; I2; . . . ; IN is the interrupt processing order. The real-time interrupt latency is related to the waiting time and the interrupt processing time. The interrupt processing is divided into the distribution and the interrupt service parts for quick analysis. The real-time interrupt latency for interrupt IK based on this division is shown in Fig. 3. The interrupt processing time (T_P (K)) comprises of two parts: the distribution time (T_D (K)) and the service time (T_S (K)) can be denoted as follows:

$$T_P (K) = T_D (K) + T_S (K)$$

T_D (K) is the time interval between the point of occurrence of interrupt and the point where it is served by reaching to its corresponding interrupt service routine.

(T_S (K)) is the interrupt service time corresponding to its interrupt service routine. The worst-case real-time interrupt latency, WCET (IK), is the summation of the distribution time (T_D (K)) and the waiting time.

WCET (IK) = (T_D (K)) + the worst-case waiting time

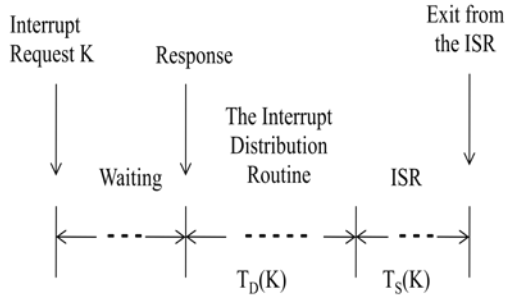

Figure 3: The real-time interrupt latency of interrupt IK...

Implementing Hybrid Operating Systems with Two-Level Hardware Interrupts [15]. It is easier to create hybrid systems with improved performance than using hardware for separating the real-time and the non-real time hardware interrupts. The significant issues for implementing a hybrid system is analysed and the RTLinuxTHIN –Real Time LINUX is implemented. In the RTLinuxTHIN-Real Time LINUX there is a two level Hardware Interrupts on the ARM architecture by combining ARM Linux kernel 2.6.9 and muC/OS-II. The results show that RTLinux THIN improves real time interrupt latencies and better predictability is provided.

3. METHODOLOGY

In this section, the different methods for controlling the dc motor are analysed using Arduino and Raspberry Pi and its description is mentioned below.

3.1. Control of Dc Motor using Arduino

The Arduino is an 8 bit microcontroller which uses L293D motor driver for controlling the dc motor. L293D is a 16 pin IC and to build this driver circuit, Pin 1 assigned for enabling and disabling the motor which comes from the Arduino digital PWM pin 9. Pin 2 is used as the logic pin for the motor ant it goes to Arduino digital pin 4. Pin 3 is one of the motor terminals can be either positive or negative. Pin 4,5,12 and 13 are ground. Pin 6 is for the other motor terminal. Pin 7 is the Logic pin for the motor to go to Arduino digital PWM pin 3. Pin 8 is the power supply for the motor.

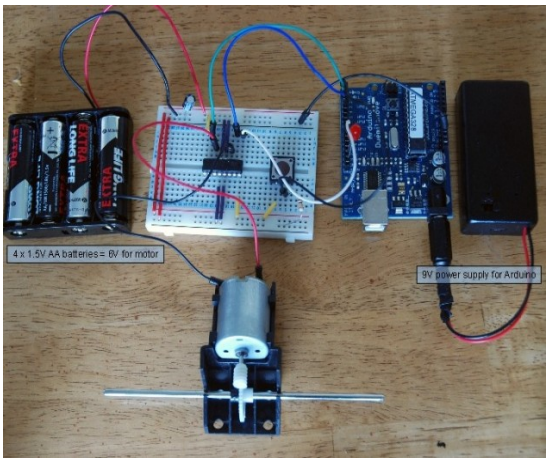

Figure 4: Image of dc motor control using Arduino and L239D chip

Pin 9 enables and disables the second motor whether it is ON or OFF. Pin 10 is the Logic pin for the 2nd motor. Pin 11 is for one of the 2nd motor terminals which can be either positive or negative. Pin 14 is the second motors other terminal. Pin 15 is the second motor logic pin. Pin 16 is Connected to +5V. Normally most DC motors need a added current than the Arduino board can afford. The drawbacks of this Arduino are overcome by the proposed system.

Journal of Theoretical and Applied Information Technology
30th June 2013. Vol. 52 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3.2. Control of Dc Motor using Raspberry Pi

In the proposed system the 32 bit microcontroller has been used. The installation of OS to the microcontroller become very simple and the code size is reduced to about 40-50 per cent. The enhanced features for increasingly complex algorithm are supported in this system and due to the device aggregation the cost is also reduced. The additional features of external connectivity such as Ethernet, USB, CAN, HDMI and code reuse is possible in this system. This system includes efficient real-time operations such as in medical, military applications etc. It also performs Multitasking operations. Availability of tools such as Midori, Python scratch, Python Ideal, LXT terminal is supported in this system.

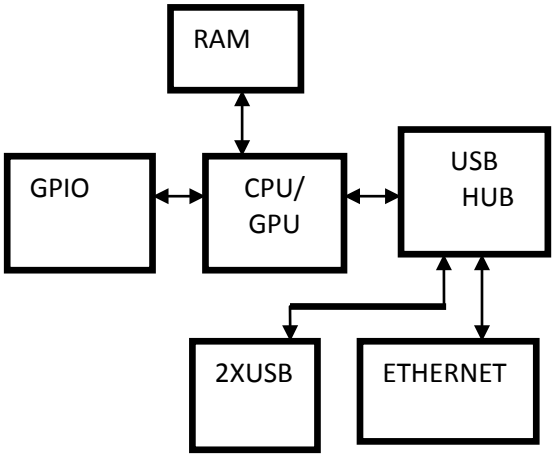

Figure 5: Block diagram of Model-B Raspberry Pi

In the Model-A the USB port is coupled directly to the SOC. The Raspberry Pi USB hub provides 3 effective USB ports, 1 of which is used by the Ethernet controller and the other 2 have physical USB ports.

Figure 6: Image of Model-B Raspberry Pi

General Purpose Input/output (GPIO) is a generic pin on a chip. At run time the behavior of the pin can be controlled by the user. GPIO pins can be configured to be input or output and can be enabled or disabled. Input values are only readable but the output values are readable as well as writable.

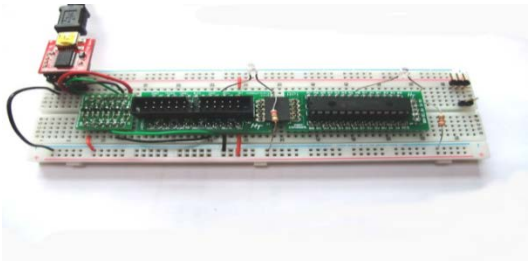

Figure 7: Picture of GPIO Connections

L293D is a dual H-bridge motor driver integrated circuit. Motor drivers receive a low-current control signal and offer a high current signal which is used for driving the motors; hence they act as current amplifiers. Two DC motors can be driven concurrently, both in forward and reverse direction in its common mode of operation. By the input logic at pins 2, 7, 10 and 15 the motor operations can be controlled. To stop the corresponding motor input logic 00 or 11 can be used. To rotate the motor in clockwise and anticlockwise directions logic 01 and 10 respectively are used. Corresponding to the two motor enable pins 1 and 9 should be made high for motors to start operating. The associated driver gets enabled as soon as the enable input becomes high. Hence the outputs work in phase with their inputs and thus become active.

Journal of Theoretical and Applied Information Technology
30th June 2013. Vol. 52 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

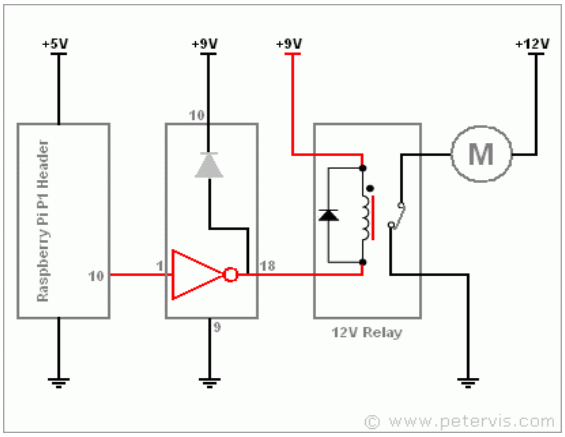

Figure 8: Raspberry Pi dc motor interfacing

In the same way, when the enable input is low, the driver becomes disabled, and their outputs are off as well as in the high-impedance. With the aid of relays otherwise some specially proposed chips the direction of the DC motor can be altered. Using a single relay for running the motor is a long way process, so another relay can be used in a reverse direction with opposite polarity.

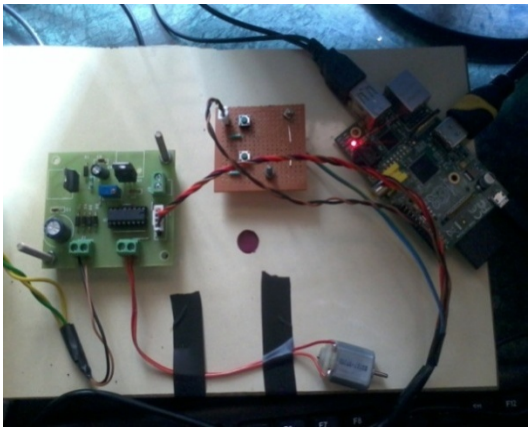

Figure 9: Image of dc motor being interfaced with the Raspberry Pi

The speed of the motor depends on three significant factors such as load, voltage, and current. For a known fixed load a stable speed can be maintained by using a PWM. The amount of power given to the motor can be increased or decreased, thus altering the motor speed by varying the width of the pulse applied to the DC motor.

4. RESULT AND DISCUSSION

Downloading the latest Raspbian distro and burning it to an SD Card ranging from 2GB to 8GB is the first step. The latest Raspbian "wheezy" distro is downloaded using www.raspberrypi.org/downloads. By choosing the .zip file the image file can be extracted. On the download page there is a link to a piece of software to burn the distro onto the SD Card. For Windows this is called Win32DiskImager and is downloaded to run. The image file for the distro and the SD Card drive is selected and the Write button is chosen.

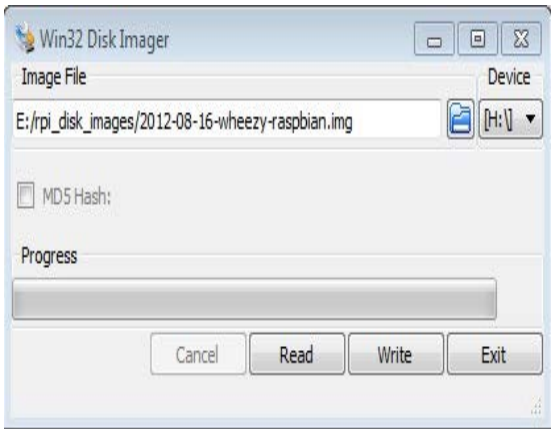

Figure 10: Image file in Win32DiskImager

Burning the image to the SD card takes time but as soon as the image is burned successfully to the SD card, that card is driven out. Then the card is loaded into the Raspberry Pi.There are two options for booting up the Raspberry Pi. Either the Raspberry Pi can be plugged into a TV or Monitor by means of the HDMI interface by attaching USB keyboard and Mouse.

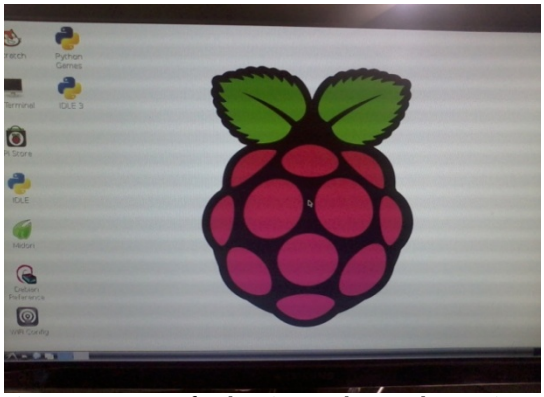

Figure 11: Image after booting up the Raspberry Pi

Otherwise the Raspberry Pi can be linked to the Raspberry Pi GPIO port's serial UART. In order to communicate with the Raspberry Pi the terminal software on a PC is used. The Raspberry Pi GPIO port can be accessed by downloading and installing the GPIO packages. The baud rate for the serial

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

port is 115200 by default. The latest version is 0.3.1. For the newer versions visit http://pypi.python.org/packages/source/R/RPi.GPIO/.The Raspberry Pi is booted up and log on using the default username and password for the Raspbian distro. The Python Development Software is installed by entering -**sudo apt-get installs python-dev.** There is a package which has command line tools for using I2C. ENTER - **sudo apt-get install i2c-tools is used to download.**

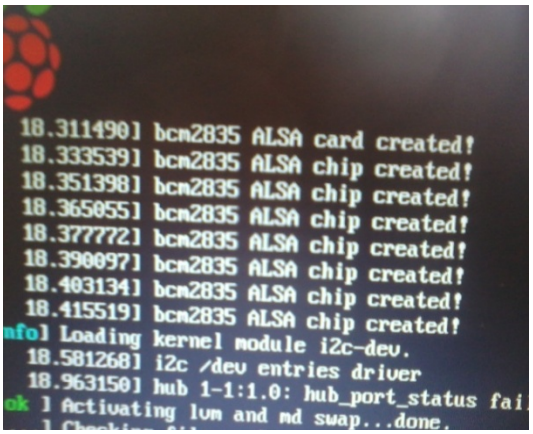

Figure 12: Image of loading kernel module i2c tools in computer monitor

Packages and modified settings have been installed. Reboot has to be done for these to take effect. To reboot the Raspberry Pi run the command- **sudo shutdown -r now**. The Pi Store is a major application of Raspberry Pi. It provides the way for sharing user's creations with a widespread audience. Some of the applications are to build a cheap laptop, tablet, create a digital photo frame, PBX, Home automation system, MP3 player, Portable personal computer, Cyber Cafe computer, Video conferencing system, Personal weather station / logger, Control of light display and LED board, CD / DVD ripping device etc.

Raspberry Pi is used to produce interrupts. To route the interrupts to the precise pieces of interrupt handling code is one of the major tasks of Linux interrupt handling subsystem. This code must understand the interrupt topology of the system. For example, the floppy controller interrupts on pin 6 of the interrupt controller then it must distinguish the interrupt from the floppy and route it to the driver's interrupt handling code. Linux uses a collection of pointers to data structures which contains the addresses of the routines to handle the system's interrupts. The routine that belongs to the device drivers in the system is responsible for each device driver to request the interrupt when the driver is started. When the interrupt occurs, Linux starts to read the interrupt status register of the system's programmable interrupt controller to determine the source of the interrupt. The Linux interrupt handling code is architecture specific as the number of interrupts and interrupt handling technique varies between architectures and systems.

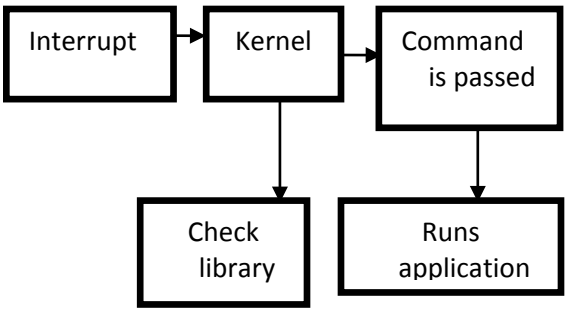

Figure 13: Hardware Interrupt Handling

When the Linux kernel calls the device driver's interrupt handling routine, it must efficiently work out the cause for the interruption and then respond. To find the origin of the interrupt the device driver would read the status register of the device that got interrupted. The device may report an error or request that the operation has been completed. The device driver may need to do additional works, once the reason for the interrupt has been determined. If it happens, the Linux kernel has mechanisms that allow it to put off that work until later. This postponing avoids the CPU from spending much time in interrupt mode.

5. CONCLUSION

A Kernel level hardware based single chip computer and device control has been presented and the advantages of this 32 bit microcontroller have been described. The Raspberry Pi can be used for many of the things that average desktop does-spreadsheets, word processing, games and also plays high definition video. The number of chip, size, and weight and power consumption is reduced. Speed and Memory of Raspberry Pi is increased. Raspberry Pi can be used in Home automation and as Media streamer too. Many modders are already trying out to join up the Pi with a touch screen.

REFERENCES

[1] Lyons, Daniel (15 March 2005). "Linux rules supercomputers". Forbes. Retrieved 22 February 2007.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[2] Schrecker, Michael. "Turn on Web Interactivity with LAMP". Retrieved 22 February 2007.

[3] Galli, Peter (8 August 2007). "Vista Aiding Linux Desktop, Strategist Says". E WEEK (Ziff Davis Enterprise Inc.). Retrieved 19 November 2007.

[4] Schofield, Jack (28 May 2009). "Are net books losing their shine?". The Guardian (London). Retrieved 2 June 2010.

[5] Shiffman, Daniel "Interview with Casey Reas and Ben Fry". Rhizome.org. , 2009.

[6] News; Arduino.cc

[7] PhillipTorrone (2011-05-12). "Why Google Choosing Arduino Matters and Is This the End of "Made for iPod" (TM)?".Makezine.com. Retrieved 2012-01-01.

[8] Miao Liu, Duo Liu, Yi Wang, MengWang, Zili Shao,"On Improving Real-Time Interrupt Latencies of Hybrid OS with Two-Level Hardware Interrupts", IEEE Journals & Magazines 2011.

[9] V. Yodaiken and M. Barabanov, "Real-Time Linux," Proc. Applications Development and Deployment Conf. (USELINUX), Jan.1997.

[10] P. Mantegazza, E.L. Dozio, and S. Papacharalambous, "RTAI: Real Time Application Interface," Linux J., vol. 2000, no. 72es, p. 10, 2000.

[11] S. Oikawa and R. Rajkumar, "Linux/RK: A Portable Resource Kernel in Linux," Proc. IEEE Real-Time Systems Symp. 1998.

[12] K.W. Batcher and R.A. Walker, "Interrupt Triggered Software Pre-fetching for Embedded CPU Instruction Cache," Proc. 12th IEEE Real-Time and Embedded Technology and Applications Symp.(RTAS '06), pp. 91-102, 2006.

[13] A. Arnaud and I. Puaut, "Dynamic Instruction Cache Locking in Hard Real-Time Systems," Proc. 14th Int'l Conf. Real-Time and Network Systems (RNTS '06), May 2006.

[14] D. Seal, ARM Architecture Reference Manual, second ed., Addison-Wesley, Nov. 2000.

[15] M.Liu, Z.Shao, M.Wang, H.Wei, T.Wang, "Implementing Hybrid OS with Two-Level Hardware Interrupt", 28th IEEE International RTSS 2007.