REMOTE DATA ACQUISITION SYSTEM USING ARM9 AND GNU/LINUX

by

ALAA ABDALHUSSAIN MASHKOR
1332320858

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science (Embedded System Engineering)

School of Computer and Communication Engineering
UNIVERSITI MALAYSIA PERLIS

2014

ACKNOWLEDGMENT

First and foremost, I would like to praise and thank Allah the almighty (SWT), who has granted me countless blessing, knowledge, and opportunity to write this project.

I offer my thanks and profound gratitude to my supervisor Professor Dr. R. Badlishah Ahmad who supported me with his experience and knowledge which has opened new vistas to me in Embedded System Engineering.

I am extremely grateful to my co-supervisor Mr. MD. Mostafijur Rahman for his sustained enthusiasm, patience, suggestions, motivation and guidance throughout the course of my research.

I am thankful to my father in law Assistant Professor Dr. al-Moussawi Ali for his support and continued encouragement to complete my postgraduate study.

I would like to give my sincere respect and appreciation for my mother and family for their support throughout the study period.

I would like to extend my thanks to my wife and colleague Zahraa Ali because she maintains a continued study and care for our children and provide everything we need in the house.

Finally, my appreciations go to all the people who have supported me in completing this research.

ALAA ABDALHUSSAIN MASHKOR

UNIVERSITY MALAYSIA PERLIS (UniMAP)

alaamh198011@yahoo.com

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

3G	Third Generation of Mobile Telecommunications Technology
ADC	Analog to Digital Converter
AJAX	Asynchronous JavaScript and XML
AM	Analysis Module
API	Application Programming Interface
APN	Access Point Name
ARC	Attached Resource Computer
ARM	Advanced RISC Machine
AT	Attention Telephone / Attention Terminal
CLI	Command-Line Interface
CPU	Central Processing Unit
DAC	Digital to Analog Converter
DHCP	Dynamic Host Configuration Protocol
DIO	Data Input Output
DMA	Direct Memory Access
DNS	Domain Name Server/Service
DSL	Digital Subscriber Line
EDGE	Enhanced Data rates for GSM Evolution
FSF	Free Software Foundation
FTP	File Transfer Protocol
GPIO	General-purpose input/output
GPL	General Public License
GPRS	General packet radio service
GSM	Global System for Mobile
GUI	Graphical User Interface
HDMI	High-Definition Multimedia Interface
HSDPA	High-Speed Downlink Packet Access
HSUPA	High-Speed Uplink Packet Access

HTML	Hyper Text Markup Language
IM	Initialize Module
IPC	Inter Process Communication
ISA	Industry Standard Architecture
ISO	International Standards Organization
LAMP	Linux Apache MySql PHP
LAN	Local Area Network
LCD	Liquid-Crystal Display
LED	Light-Emitting Diode
LKM	Loadable Kernel Module
LLC	Logical Link Control
MAC	Media Access Control
MM	Monitoring Module
NM	Notification Module
OS	Operating System
OSI	Open Systems Interconnection
PC	Personal Computer
PCI	Peripheral Component Interconnect
PHP	Personal Home Page
POSIX	Portable Operating System Interface
PPP	Point-to-Point Protocol
PPPd	Point-to-Point Protocol Daemon
RISC	Reduced Instruction Set Computer
RDAS	Remote Data Acquisition System using ARM9 and GNU/Linux
RAM	Random Access Memory
RoHS	Restriction of Hazardous Substances
SBC	Single Board Computer
SD	Secure Digital
SMS	Short Message Service

SMTP	Simple Mail Transfer Protocol
SoC	System On Chip
SQL	Structured Query Language
SSH	Secure Shell
TCP	Transmission Control Protocol
TS	Technologic Systems
TS-Linux	Technologic Systems Linux
UMTS	Universal Mobile Telecommunications System
USB	Universal Serial Bus
VPN	Virtual Private Network
WvDial	Weave-Dial

Jauh Sistem Pemerolehan data menggunakan ARM9 dan GNU/Linux

ABSTRAK

Terkini, Pelbagai Sistem Perolehan Data (DAS) boleh didapati di pasaran. yang operasi asas semua DAS adalah sama; terdapat perbezaan dari segi perkakasan rekabentuk, kadar pensampelan, resolusi, penyokong saluran perolehan data, kos, saiz, ciri-ciri mesra pengguna, dan lain-lain. Projek penyelidikan ini memberi tumpuan kepada reka bentuk dan pembangunan sistem perolehan data terbenam mencadangkan yang ciri-ciri yang menarik seperti, sokongan saluran boleubah (8-64), dua jenis pemberitahuan melalui Penhantaran Mesej Pendek (SMS) serum dan serat elektronik (EMAIL) dihantar kepada pengguna apabila menerima sebarang nilai saluran yang tidak dijangka, laman sesawang (WEB) pemantauan dan menguruskan untuk bekerja di kawasan kawalan. Seterusnya, dipanggil sistem kawalan pemerolehan data menggunakan ARM9 dan GNU/Linux (RDAS). Dalam kajian ini, fasa pembangunan terdiri daripada Papan Komputer Tunggal (SBC, TS-7800) sebagai pemprosesan unit, GNU/Linux berasas Terbenam Debian 7 Sistem Operasi (OS) sebagai platform pembangunan aplikasi dan modul kernel yang disesuaikan untuk menyokong perkakasan baru seperti modem 3G untuk menghantar pemberitahuan ke kawasan jauh. TS-7800 SBC digunakan juga sebagai pangkalan data dan pelayan WEB yang mengandungi data yang dikumpul dan boleh mengakses pangkalan data ini melalui rangkaian oleh pelayar WEB. Modul permohonan RDAS dipisahkan ke dalam Modul Permulaan (IM), yang memulakan semua peringkat keperluan modul kernel dan pengguna; Analisis Modul (AM), menganalisis data input melalui saluran ADC sama ada data kesilapan atau data sebenar dan menyimpan di pangkalan data terbenam; pemberitahuan data ralat Modul Pemberitahuan (NM) adalah memberitahu pengguna melalui SMS dan EMAIL; dan Pemantauan Modul (MM) adalah aplikasi berasaskan WEB yang mana pengguna mampu untuk meneroka status semasa dan sejarah data melalui mana-mana pelayar WEB.

Remote Data acquisition System using ARM9 and GNU/Linux

ABSTRACT

Now a days, various Data Acquisition System (DAS) is available in the market. The basic operation of all DASs are same; the differences found in terms of hardware architectures, sampling rate, resolution, data acquisition channel supports, costs, size, user friendly features, etc. This research project focuses on design and development of a proposed Embedded Data Acquisition System which interesting features are, variable channel support (8 to 64), two notification types by Short Message Service (SMS) and email send to user when receive any channel unexpected value, web monitoring system as well as manage to work in remote area; so, called Remote Data acquisition System using ARM9 and GNU/Linux (RDAS). In this research, the development phase consists of a Single Board Computer (SBC, TS-7800) as a processing unit, GNU/Linux based Embedded Debian 7 Operating System (OS) as a application development platform and customized kernel module to support new hardware like 3G modem to send notification to the remote area also using the TS-7800 SBC as database and web server which contain the data collected and can access this database through network by web browser. RDAS application module is separated into Initial Module (IM), which initiate all kernel and user level necessary modules; Analysis Module (AM), analyze the input data through the ADC channels either error data or actual data and save into embedded database; error data inform the Notification Module (NM) to notify user by SMS and email; and the Monitoring Module (MM) is a web based application by which user able to explore real and historic data status through any web browser.

CHAPTER 1

INTRODUCTION

1.1 Overview

Data acquisition is very important role in many fields such as measuring devices, processes control, engineering applications, life science research, industrial maintenance, E-government, weather station (DATAQ Instruments, 2014; National Instruments, 2104). The data collected can be used to to monitor system efficiency and ensure system reliability. This research focus to design, development and implementation of data acquisition system to get the analog data from physical parameters in a real world through variable number of channels work in parallel and save it as artificial world of digital data in embedded system. An embedded system is a specific computer system which is built into a certain system or device. Using a computer system rather than other control methods offers many benefits such small size, low cost, light weight, portable, high efficiency and low power consumption. These basic features can be used to improve the device or overall system in various ways such as reduced cost, increased dependability, improved performance, more functions and features (Dean et al., 2012).

1.2 Problem statement

All available traditional data acquisition systems in market and produced papers are: Fixed number of channels depend on ADC board used, without system notification when any channel received unexpected, almost not support or used flixable FOSS to modify application according to user need, with simple or without database to save collected data and not contain LAMP web services to monitor data from remote area.

1.3 Objectives

- To design and development of a data acquisition system supporting variable ADC channels for remote applications.

- To analyze data and send notification through SMS and Email when any channel receives unexpected data.
- To monitor current and historic data from the developed data acquisition system through any web browser.

1.4 Research Scope

This research aim to design and development an data acquisition system (DAS), with focusing on build system contains analysis data, send notification to the user by using USB modem and support variable input channels number. In this DAS used SBC connected to network to monitoring data through web browser, specially when the system placed in a remote area. The data collect from many sensors depending on analog to digital convertor (ADC) channels and save these data in database. The sensors types divided according to function of use. There are many terms and parameters used to define the performance of ADC's such as resolution, sample rate, number of channels and Quantization level. This project use embedded Linux based TS-7800 single board computer (SBC). This SBC consist ARM9 processor and many I/O interfaces to build complete system with all peripheral devices needed. Linux seems to be important in the embedded field because it's free, reliable, flexible to modify and can be upgraded to the latest version.

1.5 Thesis Outline

This work is organized as follows:

- Chapter 2 introduces the existing work and concept related to remote DAS monitoring system. It contains study of the DAS tools, embedded system and operating system.
- Chapter 3 describes the hardware components, software configuration, integration of the peripheral devices, importance of the services setup and methodology.
- Chapter 4 covers the software development, it includes creating 4 software modules, each module responsible on special manage part or more in system.

- Chapter 5 display all the system results in the form of screen shots such as kernel compiled result, connected to network by USB modem and shows the data collected through network by web browser in client side.

- Chapter 6 covers the conclusion and recommendation, it concludes the thesis by summarizing the important ideas for the contributions and future works.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

High growth of technology makes the use of The Data Acquisition system (DAS) easier and user friendly. Nowadays, researchers are more focusing on upgrading features and reducing complexity by optimizing the sampling process for particular Analog to Digital Converter (ADC) in DAS. In early stage of DAS, people use magnetic tape, paper charts and rolls as storage device. Current technology creates a wide field for the digital data storage technique (HDD, SD, Micro SD, Compact Flash) by which DAS data can be easily stored in continuous pattern, extremely accurate, error-free and reliable even though the processing speed and ADC sampling rates are different (MC Corporation, 2012). ADCs are designed either as an independent device dongle or embed with embedded computers. ADC dongle can be used as a plug and play device or need to install software to access. Some companies provide ADC peripheral card and need to integrate through PC104 or USB (Feynman, 2007). The focus point of this research is to use an embedded computer platform as a processing unit as well as ADC (either on-board or peripheral card) as a data acquisition unit to build a DAS portable and lower power consumption for using in remote area (Bakiri et al., 2012).

2.2 Data Acquisition System (DAS)

DAS, which is an important branch of embedded applications, is an integrated application of technology, based on sensors, signal measurement and data processing. Figure 2.1 shows the basic DAS components.

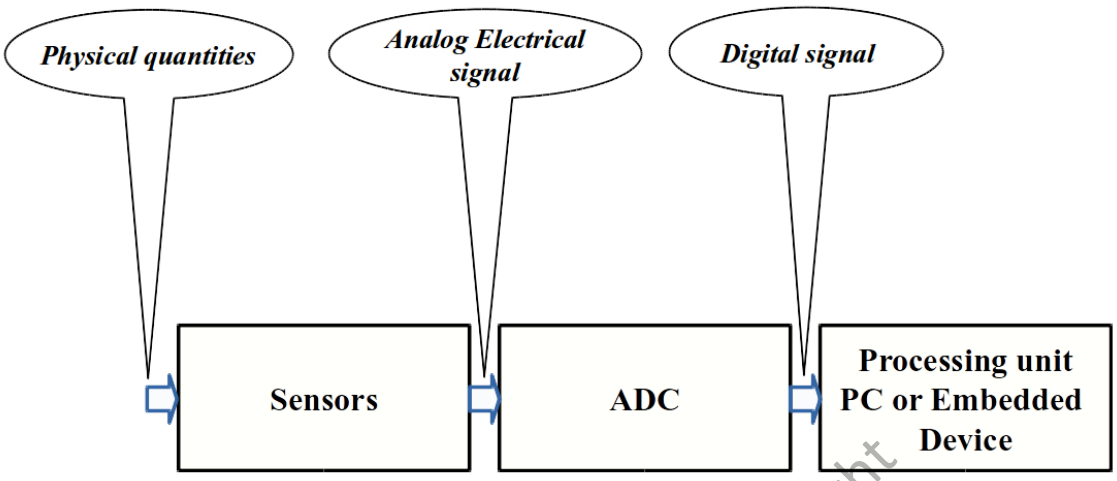


Figure 2.1: Basic DAS components

DAS is use to collect and measure physical or electrical quantities such as sound, pressure, light, voltage, current, or temperature. With a processing unit such as desktop computer system or an embedded system. The internal acquisition process is to collect analog data from a particular environment using sensors to be stored in digital form on a storage device (Scholar, 2011). A DAS, built around the flexibility of the PC and power, may contain of a wide variety of miscellaneous hardware building blocks of several equipment manufacturers. The system integrator task bring these individual components together into a full working system (John Park, 2003).

2.2.1 Sensors

There are many important variables to watch in the work environments such as temperature, humidity, lighting, voltage, length, current, smoke, pressure. To discover these variables, the physical quantities which transferred to electrical quantities by using specialized equipment. These equipments or devices called (Sensors) have many forms and types. The sensor can be divided into two main types Active sensors and Passive sensors. The Active sensors require an external source of power that provides the majority of the output power of the signal. The Passive sensors require the output power which mostly and entirely provided by the measured signal without an excitation voltage(ARC, 2012).

2.2.2 Analog to Digital Converter (ADC)

All signals found and present in the nature, are continuous and not quantized. All these signals also varied with time. Natural signals such a sound, temperature, light, are converted by a sensor or transducer to a proportional analog electrical signal. Thus, it is requisite to perform a conversion process of the analog electrical signal to a digital representation. Fundamentally, Digital signal processing systems require that signals are quantized at discrete time samples and represented as a series of words consisting of 1ŝ and 0ŝ. Typically, voltage to a binary number (Madisetti et al., 1999). This process of conversion from one format to another is called Analog to digital convertion (ADC). Where using digital circuits and digital signals offer greater advantages as compared to analog efficient transmitting and circuits in processing speed. In the conversion process there are many properties that are taken into consideration like Bandwidth, Power dissipation, Conversion Time, Sampling rate and Errors. The number of bits that represents the digital number determines the ADC Accuracy and resolution. Most ADCs, as illustrated in figure 2.2 are composed of two distinct circuit blocks performing the sampling in time and in magnitude(Halupka, 2002).

Figure 2.2: Block diagram of a typical ADC

2.2.3 Processing Unit

In DAS there is a processing unit designed to process the drawn digital streaming data. The processing unit that use in the DAS can be divided into two main types (Desktop PC and Embedded device). The PC is a general-purpose computer with standard ports, such as USB and PCI that limiting the interface devices number with this port. The embedded device designed to perform specific task by using embedded computer or microcontroller. The embedded computer is a small computer on a single board contain the microprocessor, memory and other components.

2.3 Embedded System

A system is a way of organizing, working or doing one or many tasks according to the program, set of rules or a fixed plan. Also the system arrangement in which all its units hardware and software gathering and work together according to the plan. Thus, the embedded system refer to specific computer programs that collect the functions of specially designed hardwares and softwares in one device that perform a certain tasks unlike a general-purpose personal computer. The embedded systems or computers exist everywhere such as automobiles, robotic, digital cameras, airplanes, home appliances, microwave ovens, card readers, military vehicles and equipments, medical devices, mobile communication system.

The embedded system characterized by many important properties make them more prevalent to use than other systems supported by a wide array of processors and processor architectures. They usually small size, low cost, light weight and portable, efficiency, power consumption, real-time, constraints and others (Berger, 2002).

One model of embedded system technology that uses in a recent year is a single board computer (SBC). The SBC able to perform tasks like computer as it has a processor, RAM, storage unit, I/O ports and OS. There are various SBC provider companies and selling customized depend on the application development needed for example Technologic Systems (TS)(Technologic Systems Inc, 2009), Raspberry A and B(Raspberry Pi Foundation, 2014), Beagle Bone Black(BeagleBoard.org, 2014), Parallels, Odroid-X2(Hardkernel Co., 2013), Hackberry(QuickEmbed Technology Co., 2014), UDOO(SECO USA Inc., 2014), APC Rock(APC-1, 2014), Cubie board(Cubieboard, 2014), Mars board, A13-OlinuXino(Olimex Ltd., 2014). Table 2.1 show the specifications for this SBCs. Each board is different from the other in the design, hardware and OS. TS offers a number of embedded devices with a wide variety of products, includes boot-loaders and firmware to high-level applications and operating systems.

Table 2.1: Specifications of various SBCs types

SBCs\Specs	CPU	RAM	OS	I/O ports	Storage Unit
Raspberry Pi	700 MHz ARM	512MB	GNU/Linux	USB, Ethernet , GPIO, UART, HDMI	SD Card
BeagleBone	720 MHz ARM	256MB	GNU/Linux and Android	USB, Ethernet	Micro SD Card
Parallella	800 MHz ARM	1024MB	Ubuntu	USB, Ethernet, HDMI	Micro SD Card
ODROID-X2	1.7GHz ARM	2048MB	ArchLinux	USB, Ethernet	SD Card
Hackberry	1 GHz ARM	512MB	GNU/Linux and Android	USB, Ethernet, HDMI	4GB flash, SD card
UDOO	1GHz ARM	1GB	GNU/Linux and Android	USB, Ethernet, HDMI	Micro SD Card
APC Rock	800MHz ARM	512MB	GNU/Linux	USB, ,Ethernet, HDMI, COM, GPIO	4GB flash Micro SD Card
Cubieboard	1 GHz ARM	1GB	Linux, Windows and Mac	USB, Ethernet HDMI, GPIO, UART	4GB flash SD Card, SATA
Marsboard	1.6 GHz ARM	1GB	GNU/Linux and Android	USB, Ethernet, HDMI	Micro SD Card
A13-OLinuXino	1 GHz ARM	512 MB	GNU/Linux and Android	USB, Ethernet, HDMI, GPIO, UART	SD card, SATA
TS-7800	500MHz ARM	128MB	GNU/Linux Debian	USB, Ethernet, DIO, LCD, PC104 GPIO, COM, RS232, RS485, JTAG, ADC	512 MB NAND flash, SD card, Micro SD Card, SATA

2.3.1 Hardware

The computer hardware systems consisting of many physical components work together with software (operating system, programs, and applications) to give general-purpose computer. The embedded system focus on run with limited computer hardware resources (little memory, CPU or microcontroller, NAND flash or SD card, small or non-existent screen or keyboard) this features make the embedded hardware device more desirable to perform specific task.

The most important feature of the DAS is reliability and accuracy, that can be achieved with use of reliable hardware instruments. SBC can work in harsh environments because

all the components of the SBC proven to the one Board directly by solder. This operation minimizing the number of connectors and sockets used are mostly prone to a mechanical failure. This SBC is different in hardware design, depending on the task that is performed, such as: CPU, Memory I/O Ports and OS. The essential difference is in the ARM processor architecture that gives important parameters to the SBC is power consumption. The SBCs are produced mainly using the processors architectures as following:

1. x86 a set of processor families began with the 32-bit Intel 80386 to last Intel Core i7, the most widespread processor architecture that supported by almost operating systems

2. Advanced RISC Machine (ARM) dominant and diffuse architecture on the embedded market and mobile for its low power demands, high performance and simplicity that supported by a many embedded operating systems like Windows CE, Linux and Android (Steve, 2000).

3. Performance Optimization With Enhanced RISC - Performance Computing (PowerPC) this architecture used for a long time by Apple which supported by almost operating systems that designed for this architecture specially.

The ARM processor has many features compared with X86 when used in embedded system. This includes RISC, low power, high performance and price. Table 2.2 looks at the characteristics of Technologic Systems ARM-based SBCs products (Technologic Systems Inc, 2009).

Table 2.2: Various Technologic System SBCs product characteristics based on ARM9

Product	TS-7200	TS-7250	TS-7260	TS-7300	TS-7350	TS-7370	TS-7400	TS-7800
CPU	200 MHz ARM9	200 MHz ARM9	200 MHz ARM9	200 MHz ARM9	200 MHz ARM9	200 MHz ARM9	200 MHz ARM9	500MHz ARM9
Category	SBC	SBC	SBC	SBC	SBC	SBC	SoM	SBC
PC/104 Connector	✓	✓	✓	64 pin only	64 pin only	64 pin only		✓
on-board FPGA				✓	✓	✓		✓
RAM	32 MB	32 MB	32 MB	32 MB	32 MB	64 MB	32 MB	128 MB
opt. RAM	64 MB	64 MB 128 MB	64 MB 128 MB	64 MB 128 MB	64 MB 128 MB	128 MB	64 MB 128 MB	
Flash	8 MB	32 MB	32 MB				32 MB	512 MB
std. A/D		✓	✓		✓	✓	✓	✓
Ethernet ports	10/100	10/100	10/100	2 10/100	1 10/100	2 10/100	10/100	1 Gigabit
2 USB ports	Full-Speed	Full-Speed	Full-Speed	Full-Speed	Full-Speed	Full-Speed	Full-Speed	High-Speed
SD Card socket			1 full	2 full	1 full	1 full	1 full	1 full 1 micro
Digital I/O	20	20	30	55	3	3	20	110
RS-485	opt. full/half	opt. full/half	opt. full/half	opt. full/half	opt. full/half	opt. full/half		2 opt. full/half
COM Ports	2	2	3 or 5	10	10	10	3 TTL	10
RS-232 Console	✓	✓	✓	✓	✓	✓		✓
LCD Interface	✓	✓	✓	✓	✓	✓		✓
Keypad Interface	✓	✓	✓	✓				✓
SATA Ports								2
Default Kernel	2.4	2.4	2.4	2.4	2.6	2.6	2.4	2.6
TS-Linux	✓	✓	✓				✓	
Debian Linux	opt.CF	opt.USB	opt.SD	✓	✓	✓	opt.SD	✓
Linux Fast Bootup			avail.	1.69s	under 1.5s	under 1.5s	1.1s	under 2s
Linux-based Bootloader	opt.	opt.	opt.	✓	✓	✓	✓	✓
RoHS Compliance	✓	✓	✓	✓	✓	✓	✓	✓

2.3.2 Development Software Platform

High-level programming languages usually use for writting software that are more efficient and easier for people to use. Operating system is a combination of software that manages hardware resources of the computer and at the same time provides common services for the computer programs. Embedded computer systems must be supported by operating systems designed to be used in limited computer hardware resources. There are wide variety of embedded systems, duo to a lot of the functionality requirements of embedded OSs. Because of the many technical benefits and economic, there are robust growth in the adoption of GNU/Linux OS for embedded devices. This direction has

crossed virtually all technologies and markets. The adoption rate of Linux in embedded devices continues to grow, without end in sight. The main reasons for the growth and development of embedded Linux is illustrated as follows (Hallinan, 2006).

- Linux has emerged as a stable, high-performance, mature alternative to traditional embedded operating systems(Hallinan, 2006).
- Linux can use a shell script that is a quick method of prototyping a complex application (Cooper, 2012).
- Linux supports a great variety of applications.
- Linux is free and open-source software (FOSS), the source code is openly shared and anyone is freely licensed to copy, use, change and study the software in any way(Linux.org, 2014).
- Increasing the number of software vendors, that including practically all the independent software vendor and top-tier manufacturers, now support Linux(Greg Kroah-Hartman, Jonathan Corbet, 2008).
- Linux has attracted a whopping number of active developers, platforms, enabling rapid support of new architectures of hardware and devices(Mygind et al., 2006).

For these and other reasons, there are accelerated adoption rate of Linux in a lot of new products.

2.4 Existing DAS solutions

The internal activities of data acquisition system are almost same but the differences are found with accuracy, maximum channels number usage, tasks, sampling rate, power consumption, portability, maximum input voltage, etc. Some of these DAS are built without using any system analysis and notification or cannot used in remote area, but such approach it usually not sustainable for larger projects. A number of DAS research projects and products are given as follows:

- (Peng et al., 2009) produce a research paper to the Development of the Remote I/O Data Acquisition System Based on Embedded ARM Platform, This system can measure all kinds of thermal and electrical parameters. The invention focuses on the following features:

  i) 24 channels.

  ii) Maximum analog input voltage 5v.

  iii) 16-bit ADC resolution.

- (Li & Liu, 2010) produce an research paper to Design of Embedded Data Acquisition and Remote Control System Based on Linux, with Sqlit as databases.

- (Haibo et al., 2010) produce a research paper to develop Embedded data acquisition system using microelectronic technology and the measurement technology with following features:

  i) 30 input channels.

  ii) Maximum input 3.3 volts.

- (Tiwari, 2012) produce a research paper to design Low Power, High Speed Dual, Data Rate Acquisition System using FPGA,The invention focuses on the following features:

  i) One input channel.

  ii) 16 bit resolution.

  iii) 2.5v input.

- NI DAQ: refer to National Instruments DAQ. This Product is peripheral device connect with computer to work, each module work with specific task and need to create database (National Instruments, 2104).

- ADC-20: ADC-20 indicate this peripheral device able to convert analog to digital with 20 bits resolution and need to connect with PC through USB to work, maxi-

mum input voltage 2.5 volts, 20 resolution, 8 input channels (Pico Technology Ltd, 2013).

- USB-201: peripheral device, 8 analog inputs, 12-bit resolution, Supported Operating Systems: Linux, and Mac platforms, Windows 8/7/Vista/XP, 32-bit or 64-bit; Android 3.1 and later supported by the DAQFlex framework (MC Corporation, 2014).

- DI-145: peripheral device, connect to PC through USB cable, 8 channels, max 5 volts input, 10 bits resolution (DATAQ Instruments, 2012).

2.5 Summary

This chapter has provided the insight into the overall landscape and characteristics of data acquisition and embedded systems, also the purpose for using the hardware and the software. There are many advantages to use embedded systems for DAS and monitoring system such low cost, more reliable, small size, easier to manufacture and low power consumption.

CHAPTER 3

HARDWARE AND SOFTWARE CONFIGURATION

3.1 Overview

This chapter display the design and development of the Remote Data Acquisition System using ARM9 and GNU/Linux (RDAS). The system design includes the list of all the resources involved, architecture, relevant technologies, network communication, interfaces with control centers and the application interface. RDAS is divided into three parts; hardware, software development and configuration. The hardware part provides a brief overview of the devices and components of the system. The configurations presented the system preparation and settings of embedded operating system to connect with peripherals. The system software describes the OS, programs and applications that provides an interface to user space.

3.2 RDAS Overview

The RDAS collect data from a number of channels work in parallel, to categorized and arranged in order of precedence and importance in the database and can display data by using a web browser. From this process can deduce the need to work combination of techniques with each other. These techniques and how to connect, work, implement will be detail in this chapter with all obstacles and problems solution faced by the start of the sensors and conversion from analog to digital, collecting information and create a database. On the other hand can add a special part for the alarm sends SMS and email in the case of an emergency or unexpected data received. Figure 3.1 present the overall RDAS architecture.

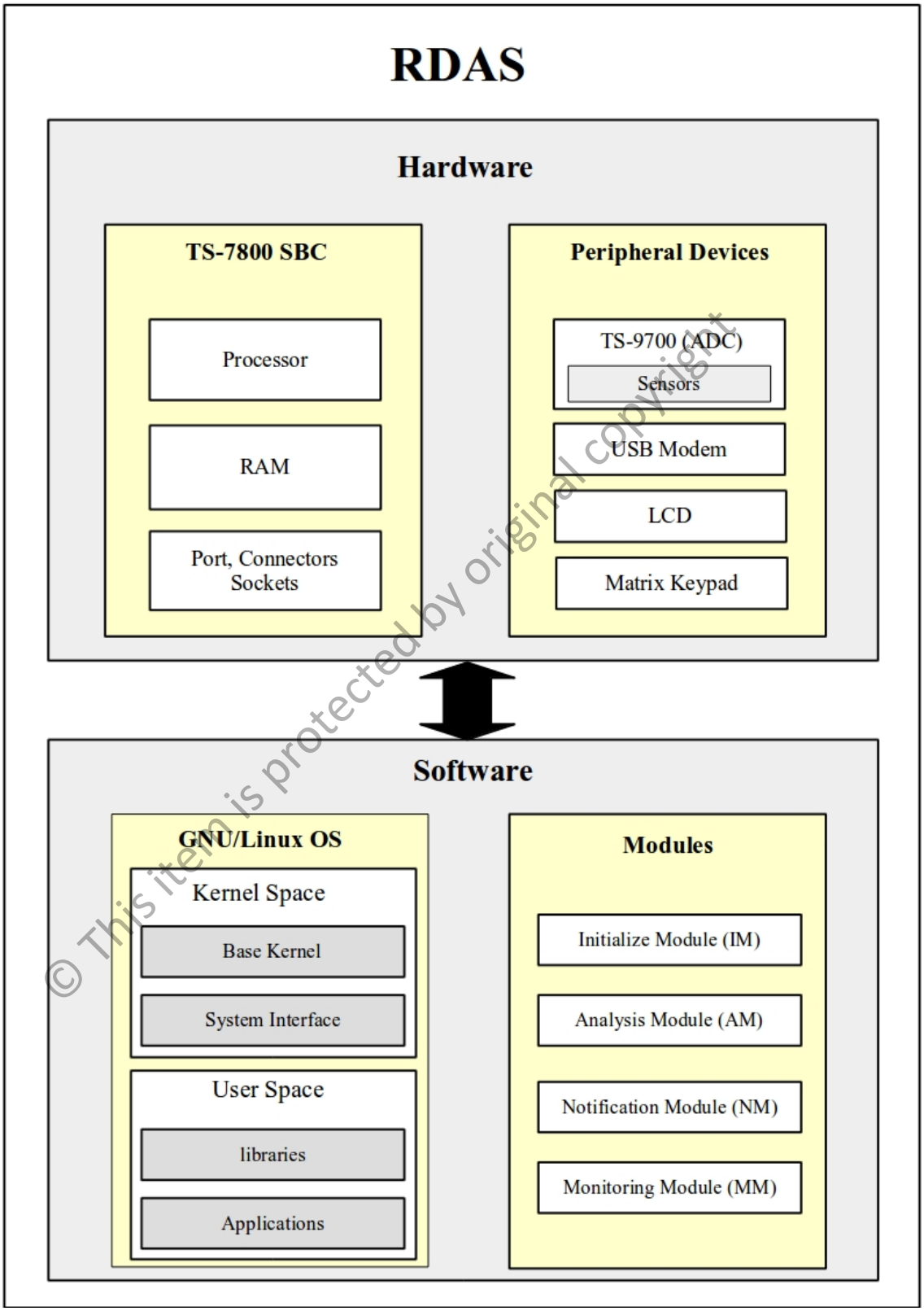


Figure 3.1: Overall RDAS architecture

3.3 Hardware Design

The hardware platform and the interrelationships can describe in simple components, begin with essential part represent by TS-7800 SBC which contains the computer system and I/O interfaces. Figure 3.2 shows hardware block diagram, TS-9700 peripheral device supports ADC that connected to TS-7800 SBCs by PC104 port and huawei modem throgh USB port. TS-9700 ADC supports 8 channel to acquire data from the sensors. The channels can be extendable by adding more TS-9700 board in parallel through PC104.

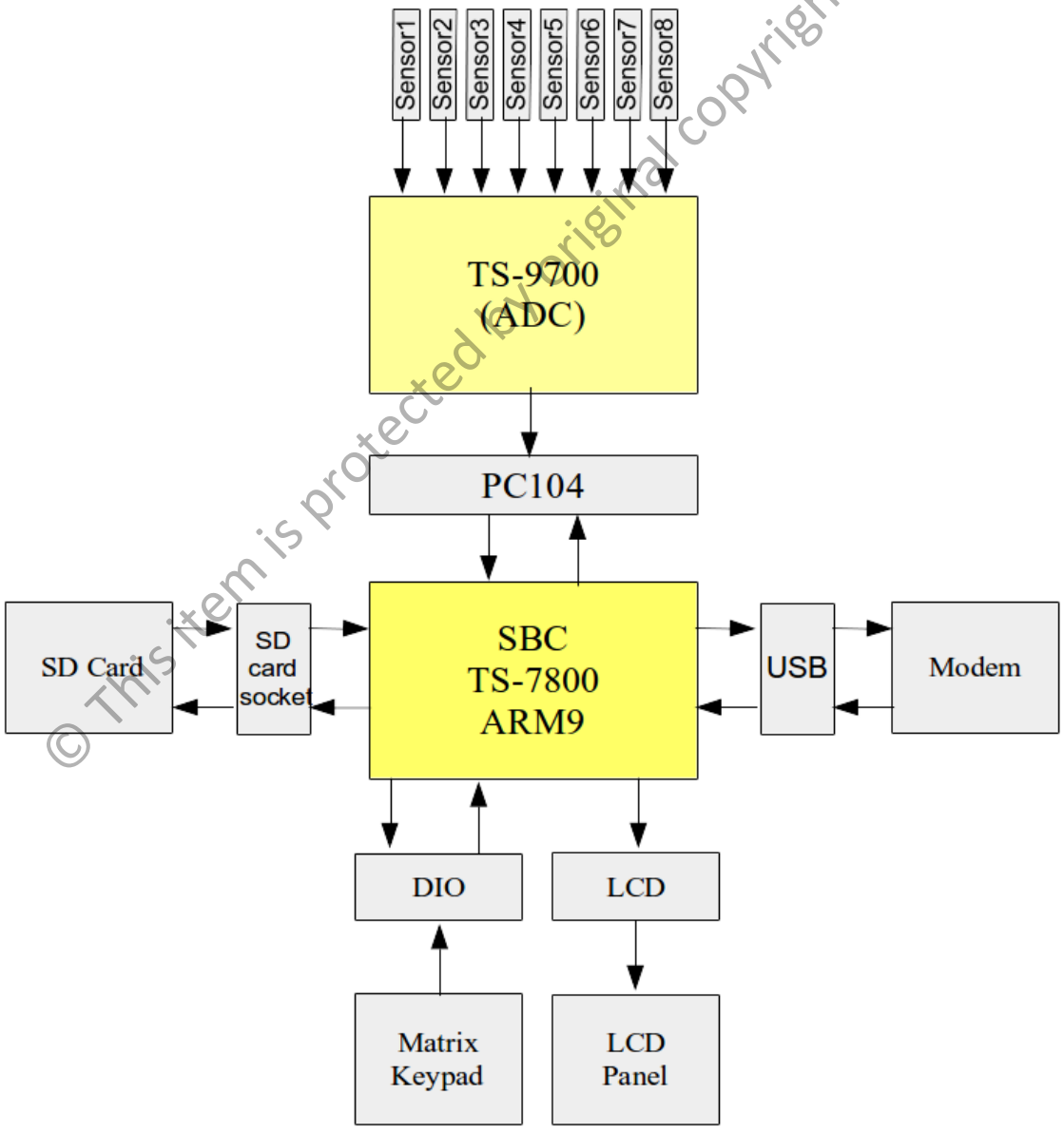


Figure 3.2: Block diagram

3.3.1 TS-7800

The TS-7800 Single Board Computer (SBC) embody one of the best available choice. The primary advantages of SBC versus Desktop PC; represented by many I/O interfaces, low-cost, low-power consumption, small-size, robust and fully integrated. PC compatible single board computer, which employs advanced embedded technology to develop compact computing systems. The SBC is a full computer built on a one circuit board. For the time being, it has been widely used in consumer electronics industrial control, navigation equipment, medical instrument, network communication, automobile, military and national defense (Technologic Systems Inc, 2009).

3.3.1.1 TS-7800 Hardware Description

TS-7800 is a compatible SBC based on a ARM9 CPU, that contains a standard set build on-board peripherals such as Gigabyte Ethernet, dual High-Speed USB 2.0 and dual SATA. PC compatibility allows for rapid development using standard development tools like Turbo C or Power Basic or GNU/Linux based tools as well. TS-7800 SBC suitable to use in build applications for a small purpose that does not require a mouse, keyboard, video card or hard drives. Figure 3.3 shows hardware components of TS-7800 SBC.

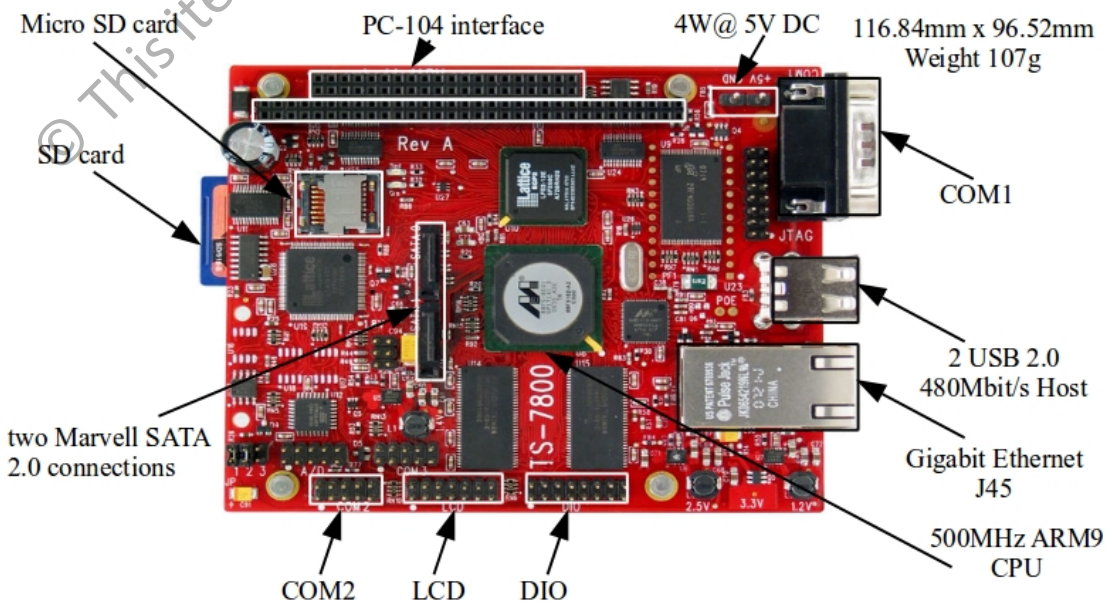


Figure 3.3: TS-7800 SBC

PC104 is an embedded computer standard port intended for specializing embedded computing environments that provide applications depend on the reliable data acquisition. The Figure 3.4 below illustrate four rows labeled A, B, C, and D for PC104 connector pins configuration.

D					19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00							
C					19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00							
A	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01
B	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01

Figure 3.4: PC104 Pins configuration

The PC104 connector can perform multi functions including Industry Standard Architecture (ISA) bus and General Purpose Input/output (GPIO). The default case is GPIO mode, with all I/Os in a neutral state. To change case by enable the PC104 bus (ISA) signals, it is requisite write values in table 3.1 to the registers specified:

Table 3.1: PC104 address

PC104 Pins No.	Row Pins No.	Row Name	GPIO	ISA
64 Pins	32	A	0x55555555	0xE8000030
	32	B	0x55555555	0xE8000034
40 Pins	20	C	0x55555	0xE8000038
	20	D	0x55555	0xE800003C

To read data from TS-9700 through PC104 bus it is required to add the base address from the table 3.2 illustrated below to the offset of the peripheral to reach a memory address for accessing the TS-9700 board. For TS-9700 ISA 8-bit I/O address 0x160, add 0xEE000000 to 0x100 to get 0xEE000160 if connect one board through PC104. When access multi-boards on PC104 the address different depend on peripheral offset address board.

Table 3.2: PC104 base address

	Memory	I/O
8-bit	0xEC000000	0xEE000000
16-bit	0xED000000	0xEF000000

For the user TS-7800 provide two USB interfaces. These are directly connected to the processor, which combine an USB dual-port Open Host Controller Interface (OHCI) with full-speed serial communications ports and hi-speed Root HUB (Technologic Systems Inc, 2009).

The standard TS-7800 SD card divided to the following partitions.

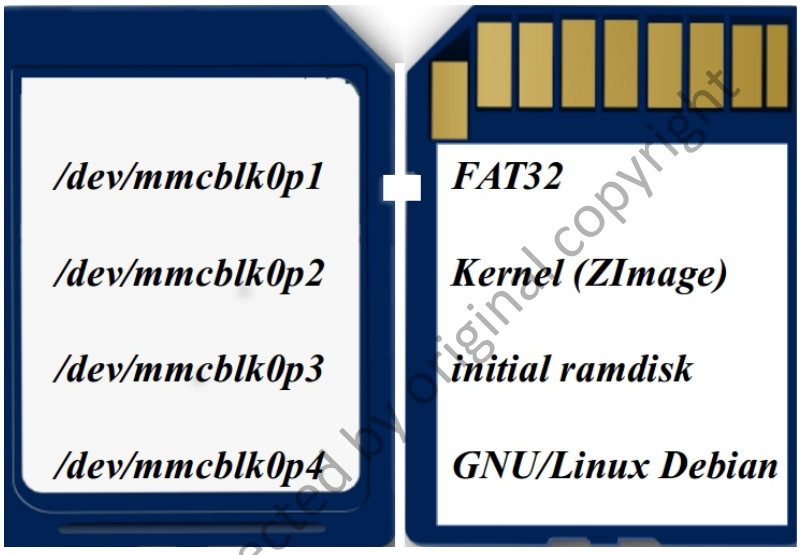


Figure 3.5: TS-7800 SD card partitions

The important partitions to boot from SD card: partition 2 contain the binary kernel image and partition 3 contain busybox and shell scripts. Then if remove or modify partition 2 or 3 the SD card unable to function properly because Master Boot Record (MBR) cannot find the address to begin.

3.3.1.2 TS-7800 Software Description

TS-7800 SBC operating system (OS) is Debian GNU/Linux 5.0 Lenny distribution on 512MB NAND Flash on-board resource and SD card. Debian 5.0.0 was initially released on February 14th, 2009 and has been superseded by Debian GNU/Linux 6.0 Squeeze distribution on Feb 06th 2011. The Lenny updates have been discontinued as of February 6th, 2012 (Debian, 2014). The latest Kernel (ZImage) on the TS-7800 Official website is version 2.6.34 with size 1.3MB support specific tasks and devices. This Kernel version is

not support the using of USB Modem. In this case, it requires to compile a new Kernel. This project collect data from a many sensors after converting data from analog to digital in the database which increase the demand to use a large storage unit, In other words, cannot use NAND Flash on-board because the size is 512MB more than 480MB used by OS but can direct use SD card to save data from any sensors numbers for long time.

3.3.2 TS-9700 Peripheral Board

The TS-9700 is a PC104 peripheral board that provides 8 channels with 12-bit ADC resolution. In the I/O space TS-9700 requires a block of 8 bytes reserved via 3 jumpers (JP1, JP2, JP3) where the address location of this block can be selected as shown in Table 3.1. This allows to using a single system with multiple TS-9700 boards. Figure 3.6 shows the PC104 and ADC on TS-9700 peripheral board.

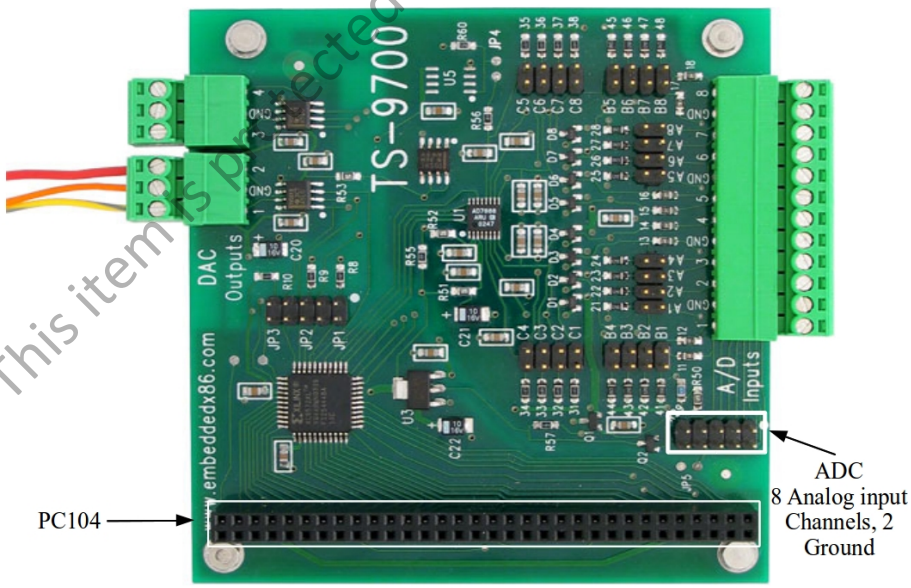


Figure 3.6: TS-9700 peripheral (ADC)

Table 3.3: Base I/O Address Selection

Jumper JP1	Jumper JP2	Jumper JP3	Base I/O Location
No	No	No	160 Hex
Yes	No	No	168 Hex
No	Yes	No	180 Hex
Yes	Yes	No	188 Hex
No	No	Yes	250 Hex
Yes	No	Yes	258 Hex
No	Yes	Yes	260 Hex
Yes	Yes	Yes	268 Hex

Table 3.3 explain the address for number of the TS-9700 ADC can connected to SBC and how can control on eight ADC boards address by 3 jumpers. Each address base reserve one byte for example if connect one TS-9700 peripheral board (all jumpers off) to TS-7800 that means the new theoretically base address for SBC it will be:

$$\begin{aligned} new\_base\_address &= (TS-7800\_base\_address) + (TS-9700\_base\_address) \\ &= 0xEE000000 + 0x160 \\ &= 0xEE000160 \end{aligned}$$

This address can use in C programing pluse the function address in HEX to recieve data from channels. Practically with these jumpers setting (all jumpers off) one TS-9700 peripheral board connect to TS-7800 SBC also and can check the PC104 address to recieve data (ADC) by LED as following.

- To enable PC104 can run the following command.

```
ts7800 :  # .  /initrd/ts7800.subr && pc104on
```

- By using LED with connect cathode to any ground port on SBC and the anode to check address in PC104 line A then note the following result.

32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01
G	0	1	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0		1									1
	LSB TS-9700 Binary Base Address (20 Bits) HSB																				AEN	IORDY	Data Channels								IOCHK

Figure 3.7: Line A from PC104 port

When convert the Binary address to Hexadecimal the result is

$101100010B = 162_{\text{H}}$

$ADBASE + ADDATA = 0x160 + 2$

$= 162_{\text{H}}$

3.3.3 Huawei E303 Modem

The E303 is a USB 2.0 high speed modem work with operating frequency for GSM,3G,4G. Interface with TS-7800 USB to access the Internet through a wireless network. Figure 3.8 indicate the E303 USB modem and Figure 3.9 show the shell output of the E303 specifications that includes ID vendor and ID product by using $lsusb$ command line.

Figure 3.8: E303 USB Modem

```
Bus 002 Device 003: ID 0461:4de7 Primax Electronics, Ltd webcam
Bus 002 Device 005: ID 12d1:1506 Huawei Technologies Co., Ltd. E398 LTE/UMTS/GSM Modem/Networkcard
Bus 002 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Figure 3.9: E303 Specifications

3.3.4 LCD Panel

Figure 3.10 is a conventional LCD panel can easily be connected with the TS-7800 as an output device. It has 48 display character spaces in two lines. Its size is 118x36x12.7mm (LxWxH), viewing area is 93.5x16mm (WxH), and character height is 3.2x5.55mm (WxH). Figure 3.10 shows the LCD panel used for RDAS. The features of this LCD panel are: low power consumption, reliable, very long operational life, user-interface display, low cost, customization and flexibility and affordable tooling costs.

Figure 3.10: LCD panel

3.3.5 Matrix Keypad

TS-7800 is connected with a conventional matrix keypad as an Input device. A 4x4, 16-button matrix keypad is used in this system. It is connected with 16 pin onboard DIO. It allows the use of standard keyboard access routine. Matrix keypad used for RDAS to shutdown, restart, connect/disconnect 3G and PC104 ON/OFF. Figure 3.11 display the matrix keypad.

Figure 3.11: Matrix keypad

3.4 RDAS Configuration

To connect peripheral devices to TS-7800 SBC through PC104, DIO, LCD and USB ports require preparation all settings of services, libraries, kernel, I/O address, OS setup, packages and drivers. The RDAS acquire data from the sensors by TS-9700 peripheral board through PC104 port and save this data into databases. To control on RDAS by using the matrix Keypad and LCD connected, these devices to DIO and LCD ports respectively. The RDAS use E303 modem connected with USB port. However, Standard TS-7800 OS Debian 5 and Kernel version 2.6.34 not support USB modem. The Huawei modem operate with Embedded Linux kernel version 2.6.35 or later (Huawei Technologies Co., 2013). Therefore need to compile new Kernel and adding required drivers.

3.4.1 Kernel Compilation

To compile a new Kernel on PC need to preparing the work environment, the capable of compiling a kernel for the ARM platform as shows in following steps.

3.4.1.1 Installing dependencies packages

Installing all libraries and necessary packages to build and compile the new Kernel.

```
linux a # apt-get install kernel-package fakeroot curl
autoconf binutils mercurial bison flex texinfo build-essential
automake libncurses5 libncurses5-dev libncurses5-dev
```

3.4.1.2 Building Cross Compilers by using Crosstool-ng

A cross compiler is a compiler eligible to create executable code for a certain platform from another different platform. its tools to generate executable programs for multiple platforms or embedded system. It use to compile for a platform which it is not suitable to do the compiling for some thousands of C codes because need big resources (CPU, RAM and storage unit) and long time, like TS-7800. This is useful in the embedded system where a device has extremely limited resources (European Community (Information Society & Technology), 2009). The latest Cross Compile available on official TS-7800 website is (ts7800-crosstool-linux-gnueabi-2008q3-2) Build on old gcc version 4.3 in 2008, more than 5 years ago. By using crosstool-ng can create a new cross compiler for ARM processor to use it in compile new Kernel on PC.

```
linux a # chmod 777 -R /opt
linux a $ mkdir /opt/x-tools
linux a $ mkdir /opt/x-tools/src
linux a $ cd /opt/x-tools/src
linux a $ wget http://crosstool-ng.org/download/
crosstool-ng/crosstool-ng-1.19.0.tar.bz2
linux a $ tar xvf crosstool-ng-1.19.0.tar.bz2
linux a $ mkdir /opt/x-tools/crosstool-ng
linux a $ cd /opt/x-tools/src/crosstool-ng-1.19.0
linux a $ ./configure -prefix=/opt/x-tools/crosstool-ng
linux a $ make
linux a $ make install
linux a $ mkdir -p /opt/x-tools/arm-unknown-gnueabi
linux a $ cp ct-ng /opt/x-tools/arm-unknown-gnueabi
linux a $ cd /opt/x-tools/arm-unknown-gnueabi
```

by using the below command it will display the Crosstool-NG configuration on terminal Figure 3.12 to select cross compiler setting for ARM platform.

```
linux a $ ct-ng menuconfig
```

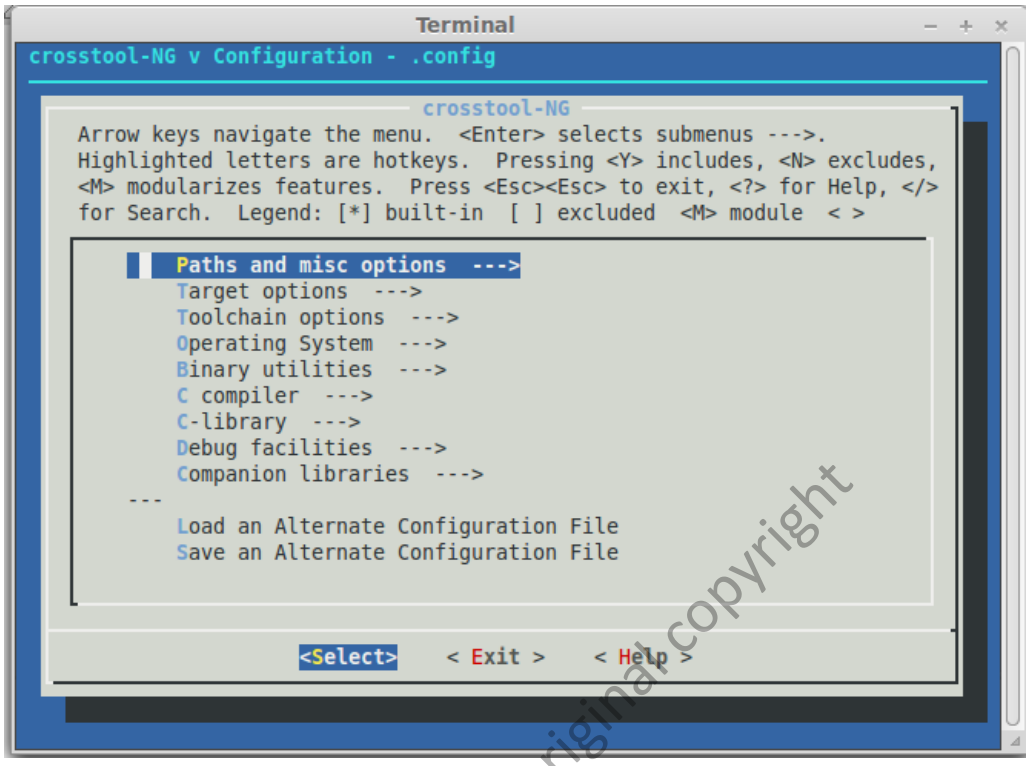


Figure 3.12: Crosstool-NG configuration

3.4.1.3 Configuration setup

1. To determine the working folder into the Paths.

((CT_TOP_DIR)/build) Working directory

(/opt/x-tools/ (CT_TARGET)) Prefix directory

2. Target option menu.

Target Architecture (arm)

Endianness: (Little endian)

(armv5t) Architecture level

Floating point: (hardware (FPU))

3. The Operating System menu.

Target OS (Linux)

Linux kernel version (2.6.37.6)

[*] Build shared libraries

Kernel verbosity: (Simplified)

[*] Check installed headers

```
(CT_TARGET)
4.  The Binary utilities menu.
Binary format:  (ELF)
Binutils (binutils)
binutils version (2.22)
5.  The C compiler menu.
C compiler (gcc)
gcc version (4.8.1)
[*] C++
6.  The Companion libraries menu.
GMP version (5.1.1)
MPFR version (3.1.2)
ISL version (0.11.1)
CLooG version (0.18.0)
MPC version (1.0.1)
```

After completed and saved configuration. Can check back, often this process will take some time.

```
linux a $ ct-ng build
```


When Cross Compile is built completely the result in (arm-unknown-Linux-gnueabi) in x-tools folder. All new binary commands for ARM architecture are internal bin folder, can use to compile kernel.

$/opt/x-tools/arm-unknown-Linux-gnueabi/bin$

3.4.1.4 Download and Modify new Kernel source code

Kernel modules are pieces of executable C code that can be unloaded and loaded into the kernel upon demand. without the need to reboot the system they provide the functionality of the kernel. Linux Kernels is Monolithic type that encompass the CPU, memory,

IPC, device drivers, system server calls and file system management. This kernels tend to be better at multitasking and accessing hardware, because if the program requests to get information from process running or memory it has more direct line to access it ,and doesnt need to wait in a queue to get things done. Figure 3.13 shows Linux kernel commutation with other system parts (Simmonds, 2010).

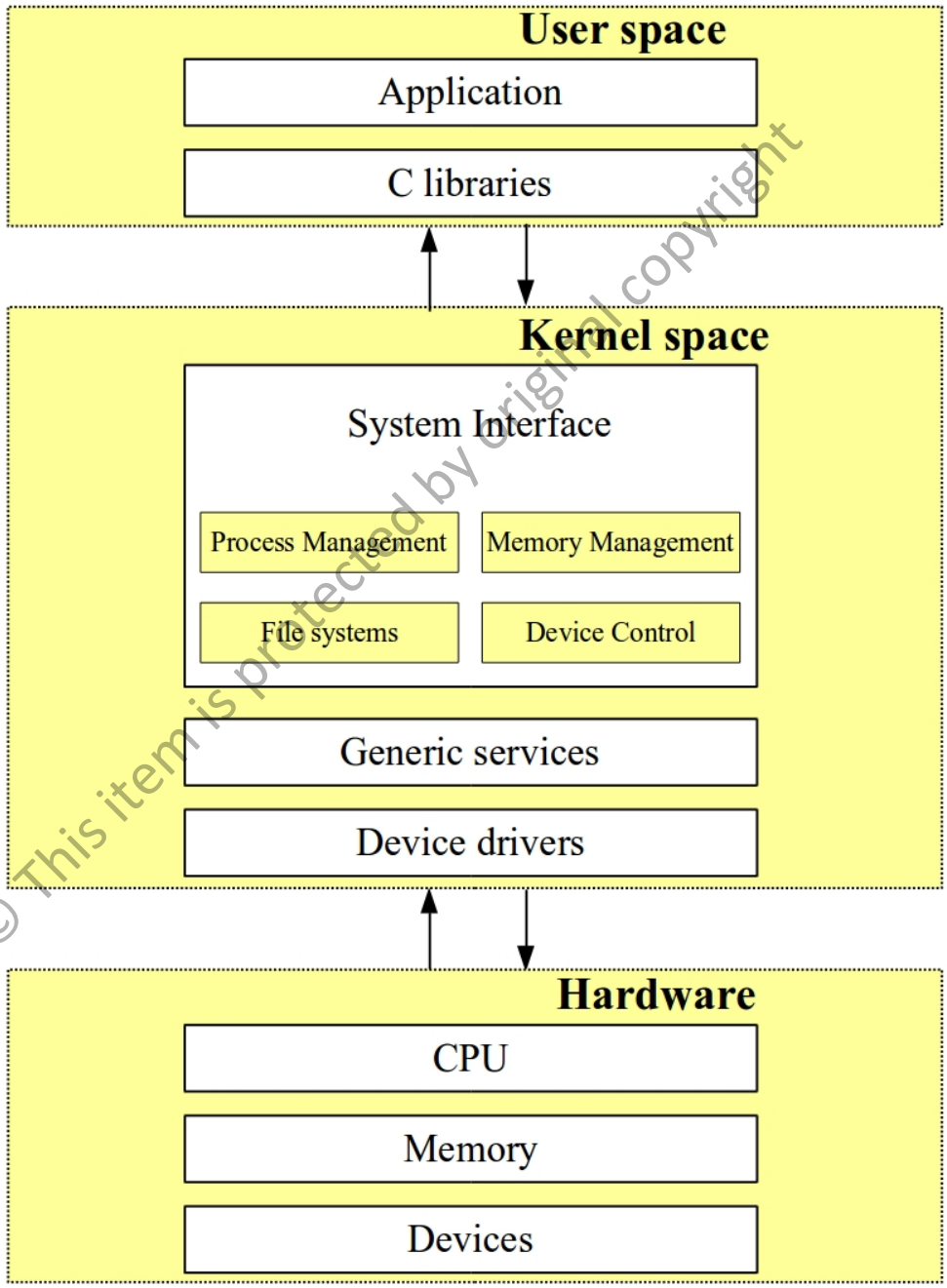


Figure 3.13: Communication between system parts

First, to connect USB HUAWEI modem to TS-7800 throgh USB port need to insert

drivers for modem in Kernel. The HUAWEI support Embedded Linux with kernel version 2.6.35 or later.

To compile new TS-7800 kernel, there are many steps can remember as following:

1- In this step need to download the TS-7800 Kernel source 2.6.34 and extract as below.

```
linux a $ mkdir /home/new-kerne
linux a $ cd /home/new-kernel
linux a $ wget ftp://ftp.embeddedarm.com/ts-arm-sbc/ts-7800-linux/
sources/linux-2.6.34-ts-src-latest.tar.gz
linux a $ tar xvf linux-2.6.34-ts-src-latest.tar.gz
```

2- To compile new Kernel require download the Kernel source 2.6.37.6 and extract.

```
linux a $ cd /home/new-kernel
linux a $ wget ftp://ftp.kernel.org/pub/linux/kernel/v2.6/
linux-2.6.37.6.tar.bz2
linux a $ tar xvf linux-2.6.37.6.tar.bz2
```

3- In new-kernel folder there are two Kernel sources

```
linux a $ ls /home/new-kernel
linux-2.6.34-ts-src-latest linux-2.6.37.6
linux-2.6.34-ts-src-latest.tar.bz2 linux-2.6.37.6.tar.bz2
```

4- TS-7800 is special-Hardware device has many I/O interfaces not available on new kernel source code. can add these drivers and libraries to new kernel from old source code after comparison between them. Figure 3.14 shows how can find the difference between two kernels version.

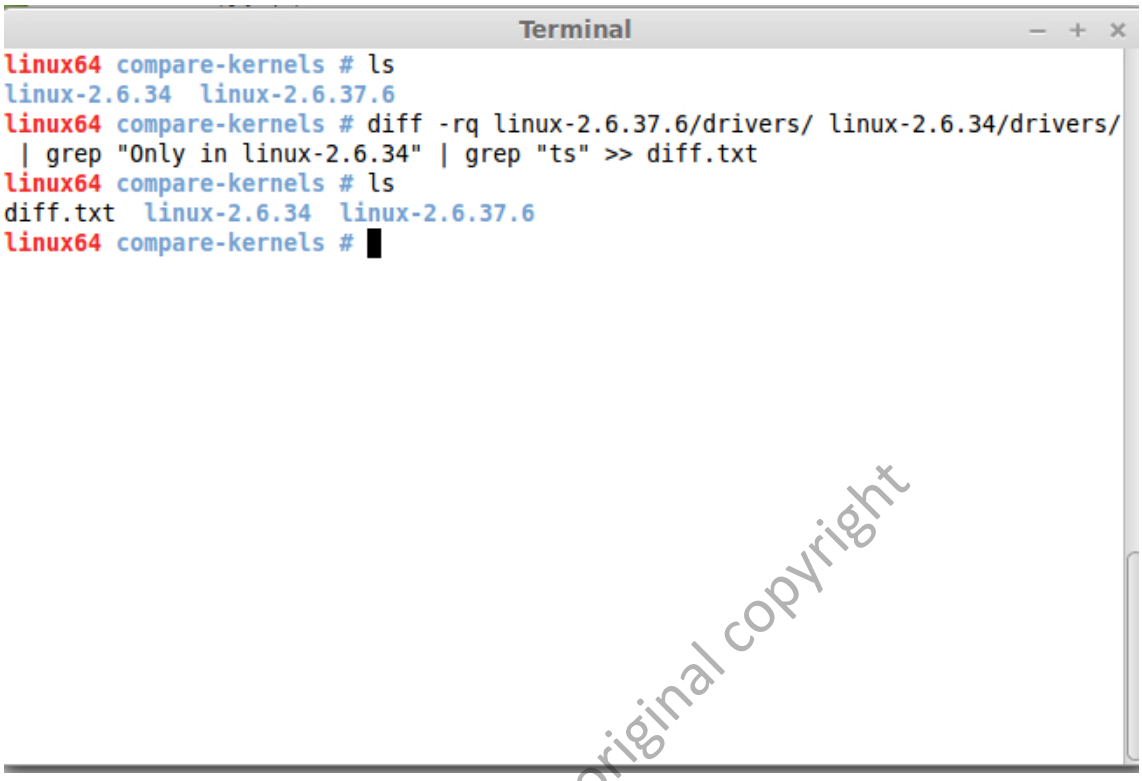


Figure 3.14: Find difference between two kernels versions


```
linux a $ cp /home/new-kernel/linux-2.6.34-ts-src-latest/
arch/arm/mach-orion5x/include/mach/irqs.h,entry-macro.S,orion5x.h
/home/new-kernel/linux-2.6.37.6/arch/arm/mach-orion5x/include/mach/

linux a $cp /home/new-kernel/linux-2.6.34-ts-src-latest/arch/arm/
mach-orion5x/irqs.c
/home/new-kernel/linux-2.6.37.6/arch/arm/mach-orion5x/
linux a $ cp /home/new-kernel/linux-2.6.34-ts-src-latest/arch/arm/
plat-orion/irqs.c
/home/new-kernel/linux-2.6.37.6/arch/arm/plat-orion/
```

5- To check revision TS-7800 by using *dmesg* and *grep* commands.

```
ts7800:# dmesg | grep rev
CPU: Feroceon [41069260] revision 0 (ARMv5TEJ)s, cr=b0053177
TS-78xx FPGA: magic=0x00b480, rev=0x0a
```

This command indicate the Board revision 10 (a in hexa) in this case to update ts78xx-fpga.h and ts78xx-setup.c (both at arch/arm/mach-orion5x/) , so the Kernel need to define line 14 of ts78xx-fpga.h

$TS7800_REV_10 = FPGAID(0x00b480, 0x0a) with line 337 of ts78xx-setup.c to case TS7800_REV_10:

6- The TS-7800 SBC access memory is not supported in new kernel source. In this case need to copy direct memory access source code from Linux-2.6.34-ts-src-latest to Linux-2.6.37.6

```
linux a $ cp /home/new-kernel/linux-2.6.34-ts-src-latest
/drivers/dma/ts7800dma.c
/home/new-kernel/linux-2.6.37.6/drivers/dma/
```

After copying dma files require to change Make file and Kconfig in drivers/dma/ folder to include the DMA stuff. By adding "config DMA_7800" part to Kconfig (can copy this part from Kconfig of 2.6.34-ts).

7- The TS-7800 SBC SD/micro SD card not support in new kernel source. To support need to copy drivers/mmc/host/tssdcore2.c and tssdcard.c

```
linux a $ cp /home/new-kernel/linux-2.6.34-ts-src-latest/drivers/mmc
/host/tssdcore2.c /home/new-kernel/linux-2.6.37.6/drivers/mmc/host/
```


Makefile and Kconfig text files need to change in drivers/mmc/host/ to include the SD card stuff. "config TS_SDCARD" part to Kconfig (copy this part from Kconfig of 2.6.34-ts).

8- To compile with default config file of TS-7800 required to copy ts7800_defconfig to new Kernel.

```
linux a $ cp /home/new-kernel/linux-2.6.34-ts-src-latest
/ts7800_defconfig /home/new-kernel/linux-2.6.37.6/
linux a $ cp /home/new-kernel/linux-2.6.34-ts-src-latest/arch
/arm/configs/ts7800_defconfig /home/new-kernel/
linux-2.6.37.6/arch/arm/configs/
```

9- The final step to complete new TS-7800 Kernel source is adding the cross compile path to Kernel Make file That created previously. And specify the Architecture is ARM at line 189 and 190.

```
linux a $ gedite /home/new-kernel/linux-2.6.37.6/Makefile
ARCH=arm
CROSS_COMPILE=/opt/x-tools/arm-unknown-linux-gnueabi/
arm-unknown-linux-gnueabi-
```

10- The new Kernel source for TS-7800 ready to config and compile. To config Kernel can use $menuconfig$ command line.

```
linux a $ cd /home/new-kernel/linux-2.6.37.6/
linux-2.6.37.6$ su
Password:
linux-2.6.37.6 # make menuconfig
```


By $make\,menuconfig$ command line can display configure window in Figure 3.15 and select Kernel configuration.

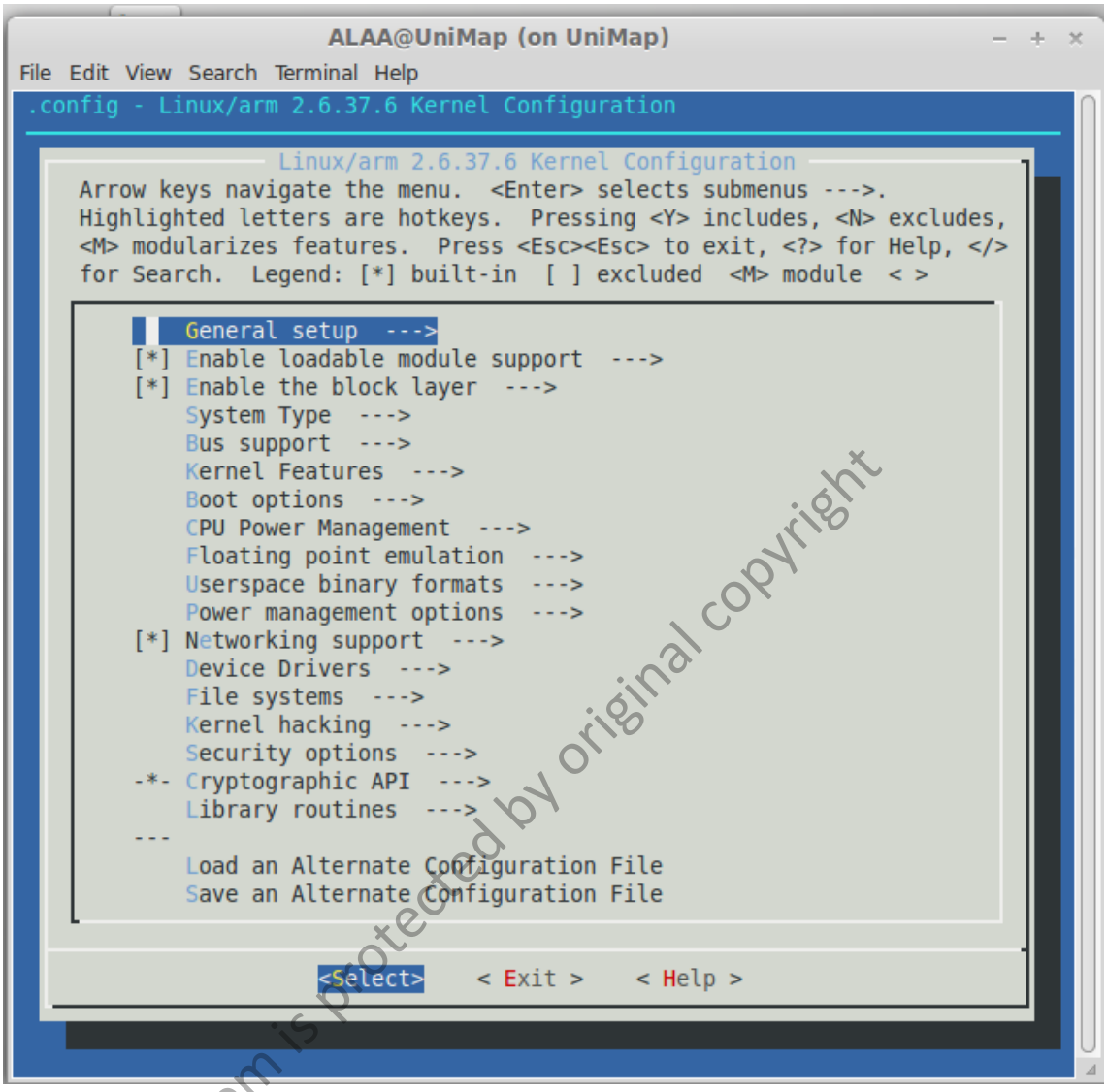


Figure 3.15: Kernel configuration

All the original settings of the TS-7800 Board found in ts7800_defconfig file, so the need to add basic new drivers non-existent as following.

i. From Power management options menu.

```
[*]Run-time PM core functionality
```

ii. From Device Drivers menu select.

```
[*] Network device support
  USB Network Adapters
```

```
    <M> Multi-purpose USB Networking Framework
     <M> Simple USB Network Links (CDC Ethernet subset)
       [*]eTEK based host-to-host cables
       [*]Embedded ARM Linux links
    <M> USB-to-WWAN Driver for Sierra Wireless modems
    <M> PPP (point-to-point protocol) support
     [*] PPP multilink support (EXPERIMENTAL)
     [*] PPP filtering
     <M> PPP support for async serial ports
     <M> PPP support for sync tty ports
     <M> PPP Deflate compression
     <M> PPP BSD-Compress compression
[*] USB support
  [*] USB runtime power management (autosuspend) and wakeup
  <M> USB Modem (CDC ACM) support
  <M> USB Serial Converter support
   <M> USB driver for GSM and CDMA modems
```

iii. From File systems menu.

```
  <*> The Extended 4 (ext4) filesystem
    [*] Ext4 extended attributes
    [*] Ext4 POSIX Access Control Lists
```

iv. After complete all configurations can be Save an Alternate Configuration File and exit.

11- To get zImage Kernel can using make command in terminal this take several time. After completed compile the new Kernel can find the zImage in below directory: $/home/new-kernel/linux-2.6.37.6/arch/arm/boot/zImage$

```
linux-2.6.37.6 # make
linux-2.6.37.6 # ls /arch/arm/boot
bootp compressed Image install.sh Makefile zImage
```

12- The file system also need to Kernel for fresh modules contain the $.ko$ files. This files can use by $modprobe$ command to insert in Kernel.

```
linux-2.6.37.6 # mkdir new-modules
linux-2.6.37.6 # export PATH=new-modules make modules_install
linux-2.6.37.6 # ls new-modules/lib/modules/2.6.37.6/kernel/
crypto drivers fs lib net
```

13- The last steps are Write new binary Kernel $zImage$ to second partition on SD card and copy essential $.ko$ files to third partition $Busy\ box$ instead the old modules in partition four (Jaan, 2013).

```
linux-2.6.37.6 # dd if=/arch/arm/boot/zImage of=/dev/mmcblk0p2
linux-2.6.37.6 # cp new-modules/lib/modules/2.6.37.6/kernel/drivers/*
/*/tssdcard.ko xuart.ko tsuart7800.ko ts7800_isa16550.ko
/media/a/725831ff
linux-2.6.37.6 # cp -a new-modules/lib/modules/2.6.37.6
/media/a/f8a84687/lib/modules/
```


3.4.2 Write binary zImage to SD Card and Resize

The standard dd image size of TS-7800 is 512MB on official Technologic Systems website, because the NAND Flash on board is 512MB. The Acquisition Data from many sensors for long time need to more space, so the SD card is the solution to get as what the user needs. The following steps explain the Write and Resize Image in SD Card.

1- Steps to download and write image to 2GB SD card commands shown below.

```
linux a $ wget ftp://ftp.embeddedarm.com/ts-arm-sbc/ts-7800-linux
/binaries/ts-images/512mbsd-lenny-2.6.34-feb-22-2010.dd.bz2
linux a $ tar xvjf 512mbsd-lenny-2.6.34-feb-22-2010.dd.bz2
linux a $ su
Password:
linux a # mkfs.vfat -I /dev/mmcblk0
linux a # dd if=512mbsd-lenny-2.6.34-feb-22-2010.dd of=/dev/mmcblk0
```

2- To resize fourth partition (Debian OS prtition), first must remove the journal from /dev/mmcblk0p4, thus turning it into an ext2 partition. Because all the changes are tracked by Journal files that have dedicate area in the file system.

```
linux a # tune2fs -O ^has_journal /dev/mmcblk0p4
```

By *gparted* command line display window in Figure 3.16 to expand ext2 partitions.

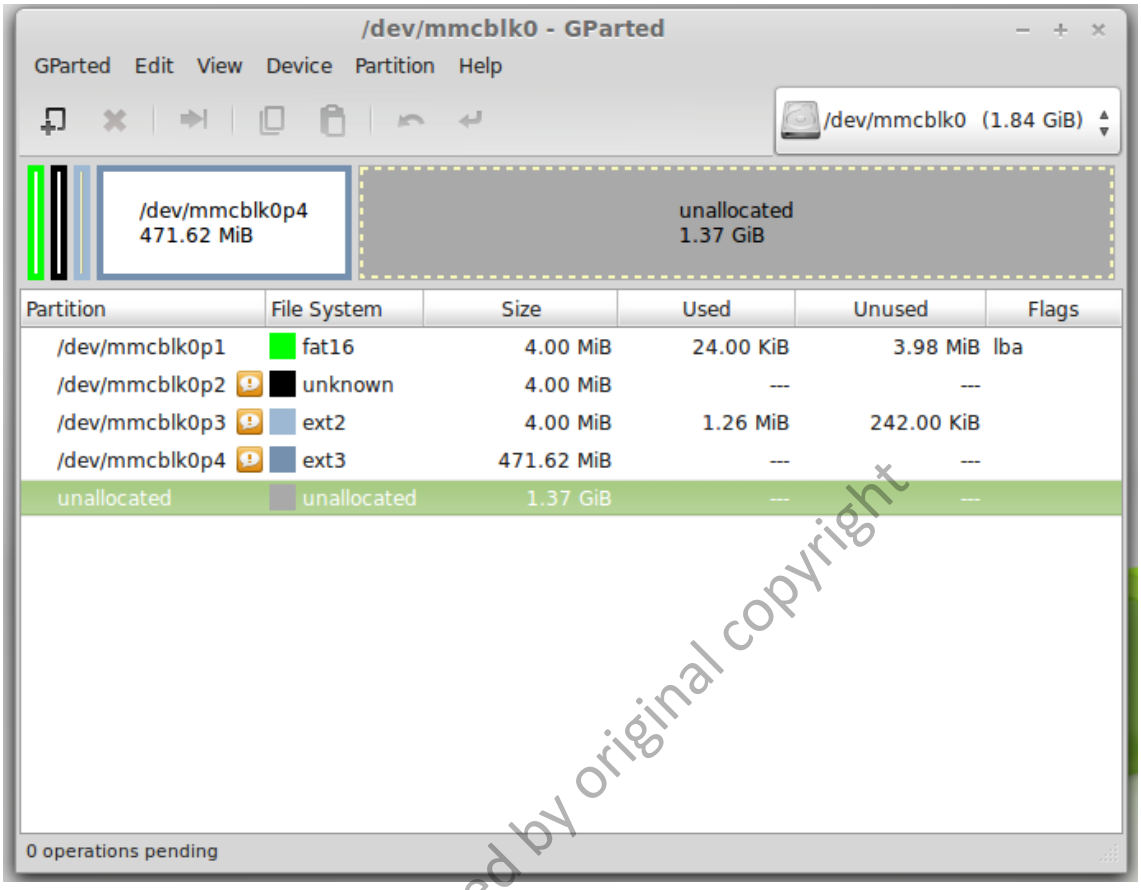


Figure 3.16: SD card partitions

Now can expand partition 4 which contain Debian OS by select it and right click *Resize/Move*. After complete Resizing write the journal file system to partition.

```
linux a # tune2fs -j /dev/mmcblk0p4
```


3.4.3 Upgrade GNU/Linux Debian OS

By the distribution maintainers assigned a priority each Debian package, as an aid to the management system of the packages. The priorities are:

Required: these packages necessary for the proper functioning of the system.

Important: these packages found on any Unix-like system.

Standard: these packages are standard on any Linux system.

Optional: these packages cover all those that might reasonably to install.

Extra: these packages conflict with others with higher priorities.

The default Debian installation whole the packages of priority important, Standard or Required will be installed in system. Moreover, some packages are marked as Essential because it is very necessary for the proper functioning of the system. The package management tools will reject to remove these (Debian.org, 2014b). To upgrade Debian 5 Lenny to 6 Squeeze need to many steps as following. 1- Can add all new Debian Repositories to sources. List in /etc/apt directory and turn off all repositories by adding # in the front of lines just keep Squeeze repository turn ON.

```
ts7800:# nano /etc/apt/sources.list
#deb http://archive.debian.org/debian/ lenny main non-free contrib
deb http://ftp.de.debian.org/debian squeeze main
#deb http://ftp.de.debian.org/debian wheezy main
```

2- To connect TS-7800 to Internet through LAN port can change the IP from static to dynamic by adding # in front each line except dhcp line.

3- The Linux tree for Squeeze can using update command to download it.

```
ts7800:# apt-get update
```


4- To upgrade Debian 5 Lenny to next distribution Squeeze Debian 6 without dependence packages problems need to remove all Optional and Extra packages by following command.

```
ts7800:# apt-get purge $(dpkg-query -Wf '$Package;-40$Priority'
| awk '$2 ~optional|extra/  print $1 ')
ts7800:# apt-get autoremove
ts7800:# apt-get autoclean
```

5- The management and responsible packages need to upgrade first from old to new Debian version to download new Linux tree.

```
ts7800:# apt-get install apt dpkg
```

6- Finally can upgrade Linux Debian 5 Lenny to Linux Debian 6 Squeeze or later version Debian 7 Wheezy by using same method.

```
ts7800:# apt-get dist-upgrade
```


3.4.4 Installing gcc

GNU C Compiler (GCC) is a system compiler created by the GNU Project can supporting C programming language. GCC is a major component of the GNU tool chain. The FSF distributes GCC under the GNU General Public License (GNU GPL). This compiler played a significant role in the development and growth of free software (Free Software Foundation, 2014).

The basic operation of gcc is converting text file in C language to executable binary file can be implemented in the Linux system. Before install gcc need to install Debian Keyring to avoid any update error.

```
ts7800:# apt-get install debian-archive-keyring
ts7800:# apt-get install gcc g++
```

3.4.5 MySQL Database

The data that collected by sensors and converted to digital data can be stored in database like MySQL or SQLite in this project used MySQL to manage users and permissions.

```
ts7800:# apt-get install mysql-common mysql-client mysql-server
```

After complete installing MySql server can create RDAS database by open MySQL and create table contain 11 columns first one for ID, second for date, third for time and others 8 for data (if one TS-9700 connected through PC104) as shown below.

```
ts7800:# mysql -u root -p
Enter password:
Type 'help;' or 'h' for help.  Type 'c' to clear the current input
statement.
mysql> show databases;
+--------------+
| Database          |
+--------------+
| information_schema |
| mysql            |
| performance_schema |
| test             |
+--------------+
4 rows in set (0.08 sec)
mysql> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
mysql> create table table0 (
   -> id INT UNSIGNED NOT NULL AUTO_INCREMENT,
   -> PRIMARY KEY (id), date char(40), time char(40),
   -> Ch0 char(20), Ch1 char(20), Ch2 char(20),
   -> Ch3 char(20), Ch4 char(20), Ch5 char(20),
   -> Ch6 char(20), Ch7 char(20) );
Query OK, 0 rows affected (0.22 sec)
```

3.4.6 Web Server

A data acquisition server is a software service which uses LAMP Server. The LAMP refers to a common combination of software used in many web servers: Linux, Apache, MySQL and PHP. Now need to install Apache Server and PHP that upload monitor Web page on it.

```
ts7800:# apt-get install apache2
ts7800:# apt-get install php5 libapache2-mod-php5
```


3.5 Installing USB Modem

Every USB device that plug into a GNU/Linux system has a vendor id and a product id that uniquely identifies the device. The usb.org given vendor id to the Supplier who in turn appoint unique product id for the products they manufacture. Using these product id and vendor id by the kernel to know whether driver supports the device or not. To find the device ID's can use the command usb-devices or lsusb. The result of command lists the details of all the USB devices that connected to any of the USBs ports, it gives information of these devices. To know the HAWEI USB modem details can type the $lsusb$ command in the terminal.

```
ts7800:# lsusb
Bus 002 Device 001:  ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 001:  ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 004:  ID 12d1:1506 Huawei Technologies Co., Ltd.  E398
LTE/UMTS/GSM Modem/Networkcard
```

The first two devices are the USB ports on the TS-7800 the third is USB modem. This modem has many functions (Internet connection, calls, send SMS and SD card reader). To find USB serial port name can using $dmesg$ command line.

```
ts7800:# dmesg | grep ttyUSB*
usb 2-1:  GSM modem (1-port) converter now attached to ttyUSB0
usb 2-1:  GSM modem (1-port) converter now attached to ttyUSB1
usb 2-1:  GSM modem (1-port) converter now attached to ttyUSB2
```

3.5.1 Connect to Internet using 3G

Most additional functions in the Linux system needs to install optional service packages to operate. The optional packages for USB modem are ppp, usb-modeswitch and wvdial. The Point-to-Point Protocol (PPP) provides a standard technique to transmit and receive datagrams over a serial link. When connecting the device to USB shows up as usb-storage by default. usb-modeswitch send command which indeed performs the switching of the device from usb-storage to usbserial WAN dongles.

```
ts7800:# apt-get install ppp usb-modeswitch
```


Each device (USB modem) need to create a Profile in /etc/usb_modeswitch.d. This profile contain DefaultVendor, Default Product for modem as shown below

ts7800:# nano /etc/usb_modeswitch.d/12d1:1506

```
# HUAWEI Modem
DefaultVendor= 0x12d1
DefaultProduct=0x14fe
MessageContent="5553424312345678000000000000000"
```

For all HUAWEI modem can define Default Vendor ID is $12d1$.

There are three ways to connect modem with Internet as following

1- Utilization wvdial

With wvdial, USB modems are automatically detect and only three additional parameters are required: the Access Point Name (APN), username, and password.

a- To install wvdial and config file in the /etc/wvdial.conf file automatically

```
ts7800:# apt-get install wvdial
ts7800:# wvdialconf /etc/wvdial.conf
```

b- In this file can add some detail manually like APN, Username and Password

```
[Dialer Defaults]
Init1 = ATZ
Init2 = ATQ0 V1 E1 S0=0 &C1 &D2 +FCLASS=0
Stupid Mode = 1
Modem Type = Analog Modem
ISDN = 0
Phone = *99#
Modem = "/dev/ttyUSB0"
Username = username
Dial Command = ATDT
Password = password
Baud = 9600
```


c- To connect 3G can type in shell wvdial command. But cannot disconnect easily, need to restart or shutdown system.

```
ts7800:# wvdial
```

2- Utilization sakis3g

Sakis3g script Manages 3G USB modems and connections. It allows to connect USB modem via the Command-Line Interface (CLI), without the need for an applet in tray. Can use this script by next steps.

a- First need to download and extract sakis3g.tar.gz as below.

```
ts7800:# wget http://jaist.dl.sourceforge.net/project/
vim-n4n0/sakis3g.tar.gz
ts7800:# tar -zxvf sakis3g.tar.gz
ts7800:# cp sakis3g /usr/bin/
ts7800:# cd /usr/bin/
ts7800:# chmod +x sakis3g
```

b- To detect Internet APN setting can enter in conf file all other necessary information.

```
ts7800:# nano /etc/sakis3g.conf
APN= maxisbb, Umobile, DiGi, celcom3g
USBINTERFACE= "0"
APN_USER= "foo"
APN_PASS= "*99#"
MODEM= "12d1:1506"
```


c- Connect and disconnect to internet network by using sakis3g connect and disconnect command respectively $sakis3g\ connect$ and $sakis3g\ disconnect$.

3- Utilization chat and pppd commands

PPP is the protocol used for Setting up internet links over DSL, dial-up modems connections, and many other types of point-to-point links. To employment this protocol need to create two files.

```
ts7800:# nano /etc/ppp/peers/3g
/dev/ttyUSB0
3600
noipdefault
debug
defaultroute
persist
noauth
nodetach
usepeerdns
connect "/usr/sbin/chat -vf /etc/chatscripts/3g.chat"
```

```
ts7800:# nano /etc/chatscripts/3g.chat
ABORT BUSY
ABORT 'NO CARRIER'
ABORT VOICE
ABORT 'NO DIALTONE'
ABORT 'NO DIAL TONE'
ABORT 'NO ANSWER'
ABORT DELAYED
"" ATZ
OK ATQ0V1E1S0=0&C1&D2
OK ATDT*99#
CONNECT ""
```


Can then run $pon\,3g$ command to connect, and $poff\,3g$ command to disconnect. When comparing among these three ways to connect 3G network I found the best way is that do not need a long time to connect after login and simple is sakis3g.

3.5.2 Send SMS by USB Modem

If TS-7800 received unexpected data value from one sensor or more it will send alert by SMS and email to user informed of the situation. There are many ways to send SMS from Linux OS by using 3G USB modem. There are a some ways to send SMS as follow.

1- AT commands

AT commands are used to control modems with Serial Communication Programs to do their specified functions. To communicate with the modem, can use minicom, picocom or screen programs. All these programs and other works in the same procedure for send command to modem.

```
AT

OK

AT+CMGF=1

OK

AT+CMGS=(Phone_Number)

> Text Message

OK
```


2- Gammu Program

Gammu is command line library and utility to work with modems and mobile phones from many vendors that support for different models. This program can work with messages MMS, SMS and EMS. It also supports to send and receive SMSs. To send SMS by gammu first need to adjust the Settings. To select the right USB port and bit rate by use the following command (Debian.org, 2014a).

```
ts7800:# gammu-config
```

The SMS can send as shown below by using $gammu$ command line.

```
ts7800:# echo "SMS text" | gammu sendsms text Phone-Number -report
```

3- gsm-utils Program

Some simple CLI programs to perform some GSM mobile phone functions via GSM modem such as sending and receiving SMS messages. To install gsm-utils and send SMS can run the following command lines.

```
ts7800:# apt-get install gsm-utils
ts7800:# gsmsendsms -d /dev/ttyUSB0 phone-number "text to send"
```

4- Chat Program

The conversational exchange between the modem and the computer can defines by chat CLI. Its basic purpose is to set up the connection between the remote's Point-to-Point Protocol Daemon (pppd) process and the pppd. Can create simple shell script to send SMS with chat command line (Gite, 2010).

```
#!  /bin/sh chat -v -t 2 AT+CMGF=1 OK < /dev/ttyUSB0 > /dev/ttyUSB0
chat -v -t 2 AT+CMGS=Phone Number > < /dev/ttyUSB0 > /dev/ttyUSB0
chat -v -t 2 Text to Send > < /dev/ttyUSB0> /dev/ttyUSB0
chat -v -t 5 Ẑ +CMGS: 1 < /dev/ttyUSB0> /dev/ttyUSB0
```


3.5.3 Send Email

There are many ways to send an email but the simple and easiest way is Simple Mail Transfer Protocol (SMTP). To use SMTP need to install two packages msmtp and mailx as following.

```
ts7800:# apt-get install msmtp heirloom-mailx
```

Configuring msmtp by creating msmtp configuration file. A file named .msmtprc in root directory, to open, edit it can add the following setting.

```
ts7800:# nano .msmtprc

account gmail1
auth on
host smtp.gmail.com
port 587
tls on
tls_trust_file /usr/share/ca-certificates/mozilla/
Equifax_Secure_CA.crt
user ********@gmail.com
password *********
from ********@gmail.com
account default:  gmail1

ts7800:# chmod 600 .msmtprc
```

To configure mailx, also can create another configuration file in root directory .mailrc. The following code goes in that file:

```
ts7800:# nano .mailrc

set from=Sender_EMAIL_ID
set sendmail="/usr/bin/msmtp"
set message-sendmail-extra-arguments="-a gmail1"

ts7800:# chmod 600 .mailrc
```

To send mail can run mailx RECIPIENT_EMAIL_ID. This will be prompted to enter Subject and Body of the mail.

```
ts7800:# mailx -s "subject" RECIPIENT_EMAIL_ID < body.txt
```

Or can use *mail* command to send Email.

```
ts7800:# echo "body" | mail -s "subject" RECIPIENT_EMAIL_ID
```

3.6 Compile and running RDAS main program

After installation and configration all necssory packges and programs can run and compile main RDAS C code. This code contains MySQL source, need to install the MySQL C development libraries below.

```
ts7800 :# apt-get install libmysqlclient-dev
```

Can compile the RDAS main program and the below sample shows using MySQL C development libraries.

```
ts7800:# gcc data.c -o data `mysql_config -cflags -libs`
```


3.7 Summary

TS-7800 SBC is used as a hardware platform with Debian Linux embedded OS. This chapter prepare all Hardware, software development and configuration to running RDAS begin from how can connect ADC peripherals and work, compiled new kernel to support USB modem, install all necessary packages(database and web server) and techniques to send SMS and email.

CHAPTER 4

SOFTWARE DEVELOPMENT

4.1 Overview

This system was developed on embedded GNU/Linux based Single Board Computer. The selection of GNU/Linux OS is to utilize the availability of open source, shell script, simple modify, lightweight, libraries, kernels and drivers in developing and implementing this system. This chapter describes the development process of RDAS. The RDAS compose of four modules: (i) Initialize Module (IM), (ii) Analysis Module(AM), (iii) Notification Module (NM), (iv) Monitoring Module (MM). Figure 4.1 shows the flowchart design for RDAS application modules. The following sections describe on the design and development of RDAS application modules.

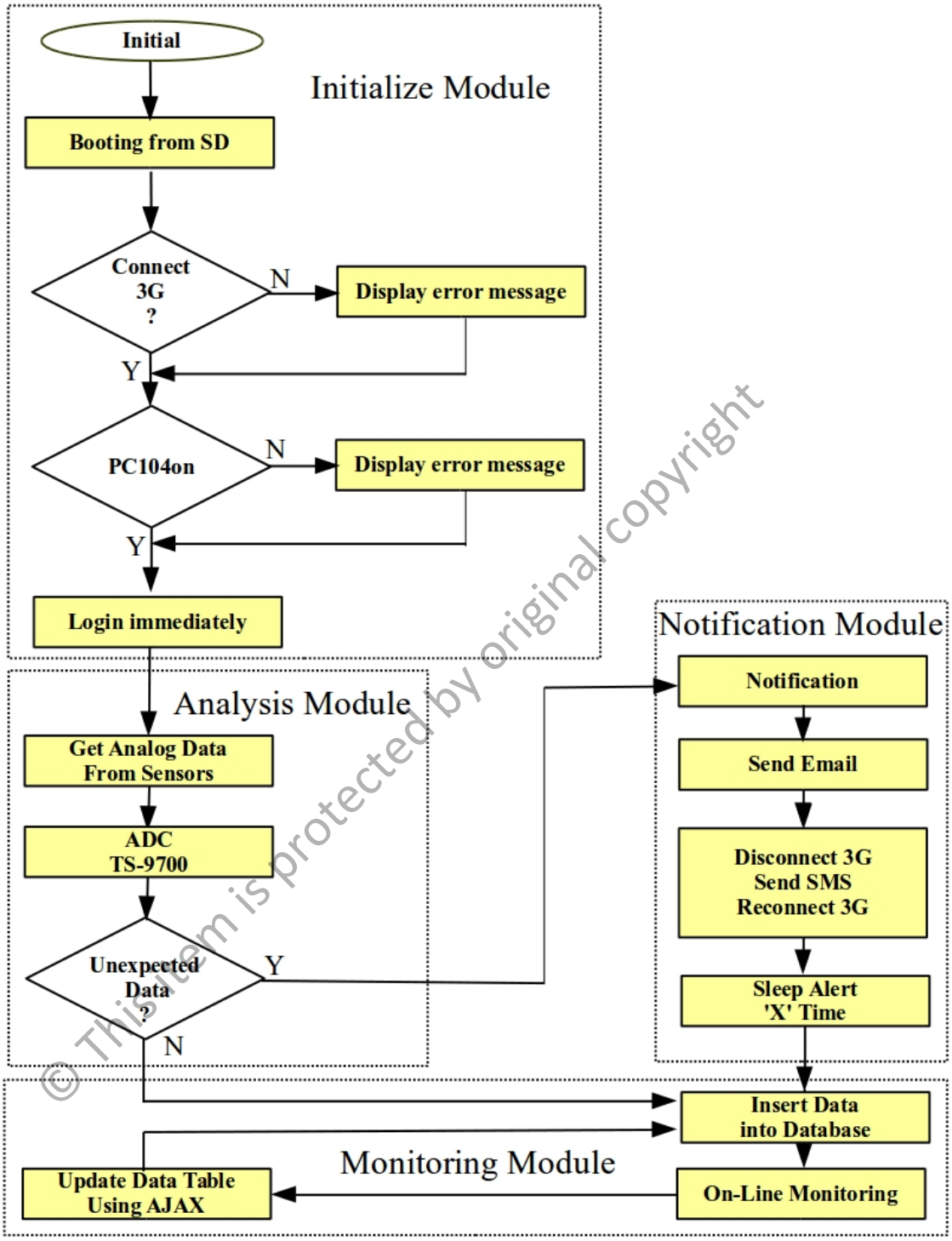


Figure 4.1: Remote Data Acquisition System Flowchart

4.2 Initialize Module (IM)

This first part of RDAS covers Linux system boot, booting from SD card, initialize all OS necessary packages, Connect to network automatically, enable PC104 port and Login

immediately no need to user name and password. In other word when connect the RDAS to the power supply it works directly without any configuration. The initial module can be divided into the following parts.

4.2.1 Booting from SD card

Collecting data from many sensors for long time need to using big capacity storage unit. TS-7800 storage options: NAND flash(512MB), SD card (max 32GB), Micro SD card (32GB). SD card choice gives high capacity space and removable storage unit. To relink to automatically boot to SD card need to run following command.

```
rm linuxrc && ln -s linuxrc-sdroot linuxrc && save
```


4.2.2 Connect to network automatically

To connect 3G automatically after login system need to create simple shell script run directly after login (Gite, 2010).

```
ts7800:# nano /etc/profile.d/auto3g.sh
#!  /bin/sh
# PATH=/etc/profile.d//auto3g.sh
echo "connecting via sakis3g"
/usr/bin/sakis3g "connect"
ts7800:# chmod +x /etc/profile.d/auto3g.sh
```

4.2.3 Enable PC104

To enable the PC104 bus signals, that is require to write the state ON values to the registers specified using simple shell script.

```
ts7800:# nano /etc/profile.d/pc104.sh
#!  /bin/sh # PATH=/etc/profile.d/pc104.sh
peekpoke 32 0xe8000030 0x55555555 peekpoke 32 0xe8000034 0x55555555
peekpoke 32 0xe8000038 0x55555 peekpoke 32 0xe800003c 0x55555
```

```
ts7800:# chmod +x /etc/profile.d/pc104.sh
```

4.2.4 Login immediately without root password

The useful of auto-login system to execute some programs or applications on boot up and call terminal commands or some scripts from within that program. This depends on to get the prompt and commands will be executed. First step is using execlp command in simple C program and compiled it then can copy the auto-login binary file to $/sbin/autologin$ (linuxgazette, 2001).

```
execlp ( "login", "login", "-f", "root", 0);
```

To complete login immediately need to change the /etc/inittab file as sample below.

```
ts7800:# nano /etc/inittab
T0:23:respawn:/sbin/getty -L ttyS0 115200 vt100
To
T0:23:respawn:/sbin/getty -n -l /sbin/autologin ttyS0 115200 vt100
-l, -login-program auto login
-N, -skip-login skip the standard login
```

4.3 Analysis Module (AM)

The analysis module responsible to get and checking the input data values. This module contains three parts (i) Sensors, (ii) ADC, (iii) Check data. The Sensors numbers depend on input channels number to get many of the environment variables as analog data. TS-9700 peripheral board convert the analog data to digital and send this data to SBC board by PC104 port to check. The maximum input data voltage through channels is 2.499 volt. So it can identify the largest and the lowest value for each sensor depending on its function as follow. If received unexpected value from any ADC channel it will gives command to send SMS and Email through notification modules. Figure 4.2 shows flowchart for analysis module.

```
if (volts= 2.499)
sprintf(buf0, "%1.3fERR ", volts);


else
sprintf(buf0, "%1.3f ", volts);
```

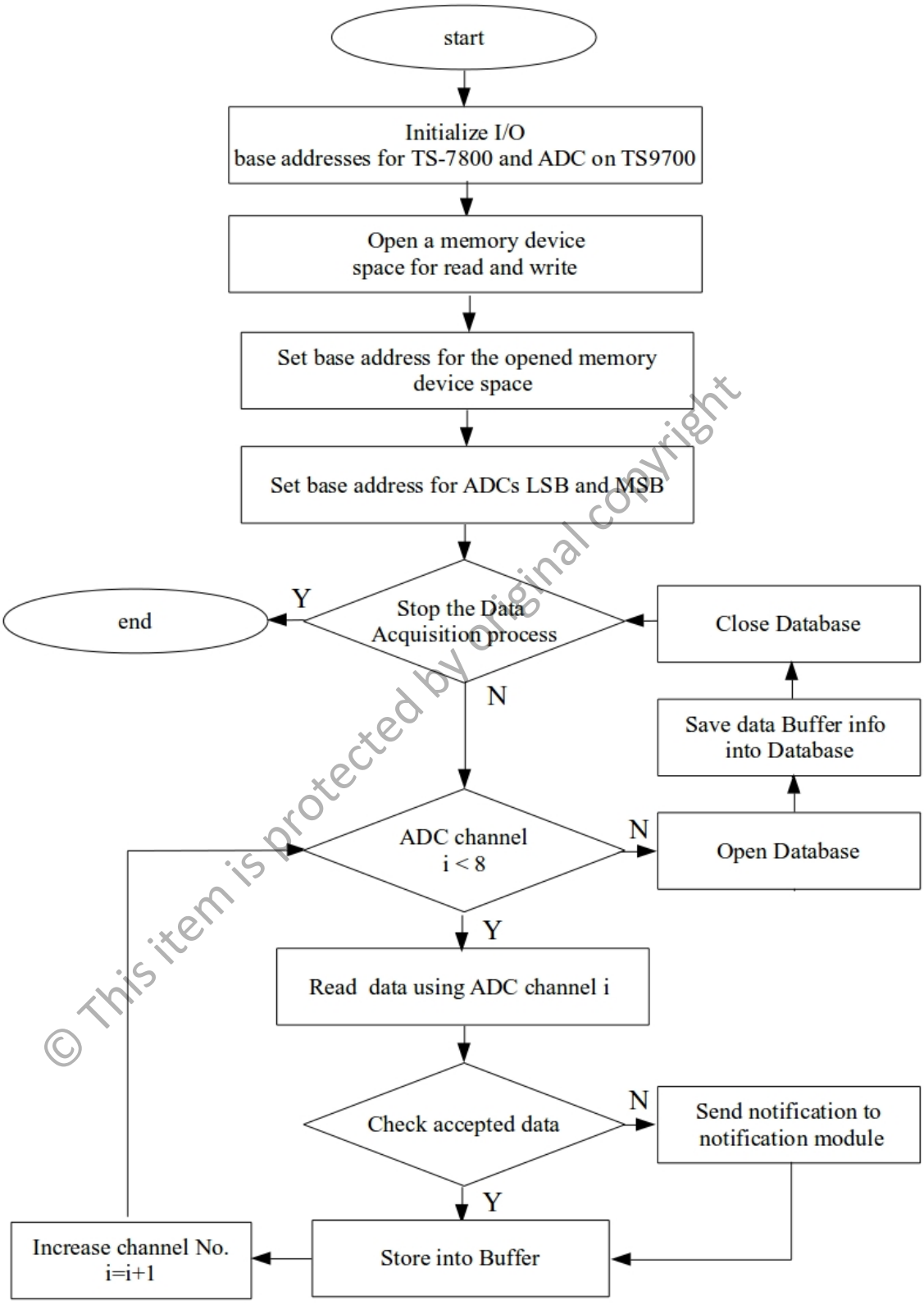


Figure 4.2: Flowchart for Analysis Module

4.4 Notification Module (NM)

When an unexpected data received from any sensor the NM will send SMS and Email alarm to inform admin there are something unnatural in the environment work. The SMS

send by using chat command, because other ways to send SMS used part from RAM and CPU. The below shell script run by RDAS main program using system command for each sensor. If AM receive any unexpected data, it will give order to NM to send SMS and Email to admin, also if the unexpected data continues to receive, the NM resend alert after 15 min to admin. The send SMS shell script show below use the AT command to give order to modem.

```
chat -v -t 2 "" "AT+CMGF=1" "OK" < /dev/ttyUSB0 > /dev/ttyUSB0
chat -v -t 2 "" "AT+CMGS=¨phone-number¨ ">" < /dev/ttyUSB0 >
/dev/ttyUSB0
chat -v -t 2 "" "Hi admin, Can Check Sensor No.'X' " ">" <
/dev/ttyUSB0>
```

The other alert shell script is to send Email to admin as a second notification.

```
echo "Hi admin, Can Check Sensor No.'X' " | mail -s "Unexpacted"
alaamh198011@yahoo.com
```


4.5 Monitoring Module (MM)

All data collected is stored in the database using MySQL and can be accessed through the use of a username and password on TS-7800 SBC board SD card. The (http://www.embeddedarm.com used a TS-7800 to host this website running Linux, Apache, PHP and MySQL. In this project can also use TS-7800 as server to host monitoring date website through web browser.

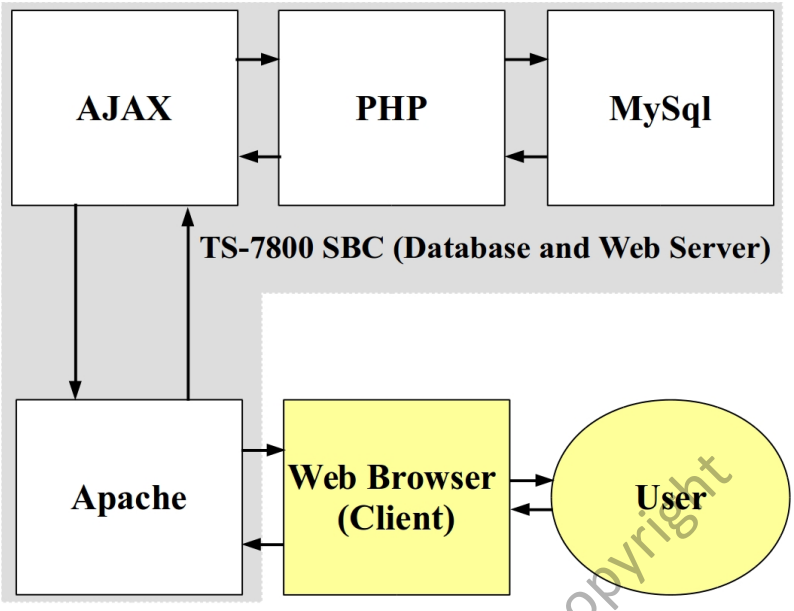


Figure 4.3: Monitoring Module sequence

4.6 Summary

The RDAS contains four modules: Initialize Module, Analysis Module, Notification Module, and Monitoring Module. The Initialize Module begin from boot, login automatically, network connection, and turn ON PC104. The Analysis Module check and analyze all coming data from channels. Notification Module send alert to inform the admin when the RDAS received unexpected data from any channels. Monitoring Module is using to monitor the database by using any web browser.

CHAPTER 5

RESULTS AND SYSTEM PERFORMANCE

5.1 Modules Results

This chapter display the overall results of system for all modules. Initialize Module (IM) focus on TS-7800 SBC board boot process (initial, login, network connection, enable PC104 and check sensors connected or no). Figure 5.1 shows shell print screen for the IM result when the SBC connect to power supplay and start boot.

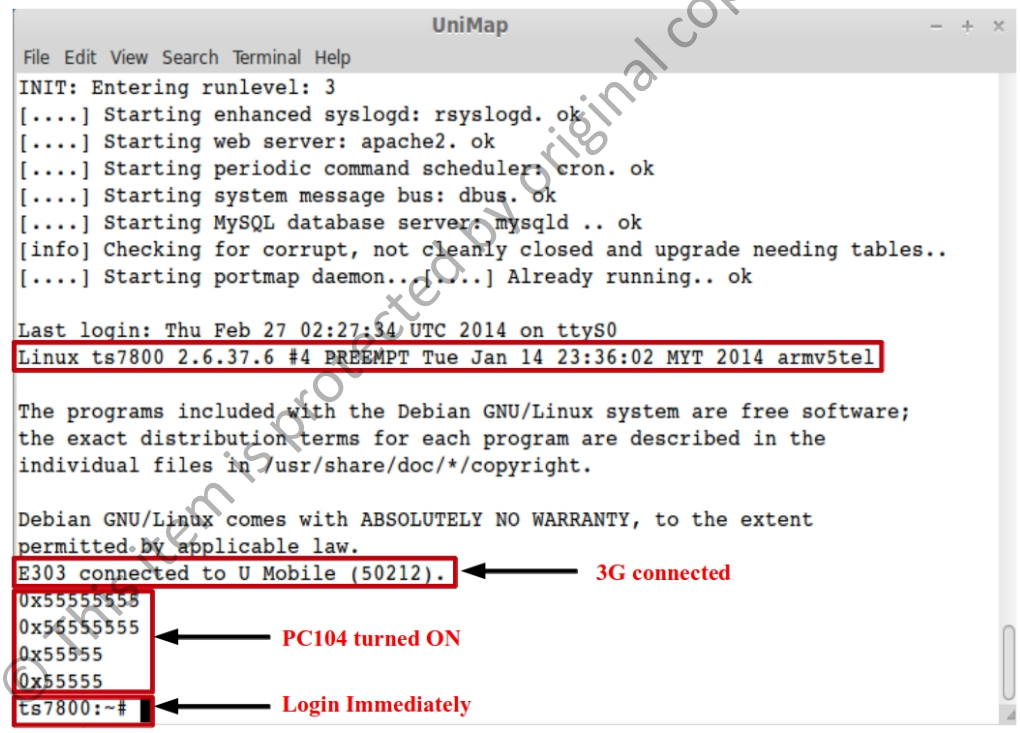


Figure 5.1: initialize module results

The analysis module (AM) works as data checker and saver. By applaying variable voltage on ADC channels this module will check the data value if unexpected data it will give command to Notification Module (NM) to send SMS and email and save data into the database if normal data value it will arrange the data with date and time in database table. Figure 5.2 and 5.3 shows the notification sent by SMS and Email respectively.

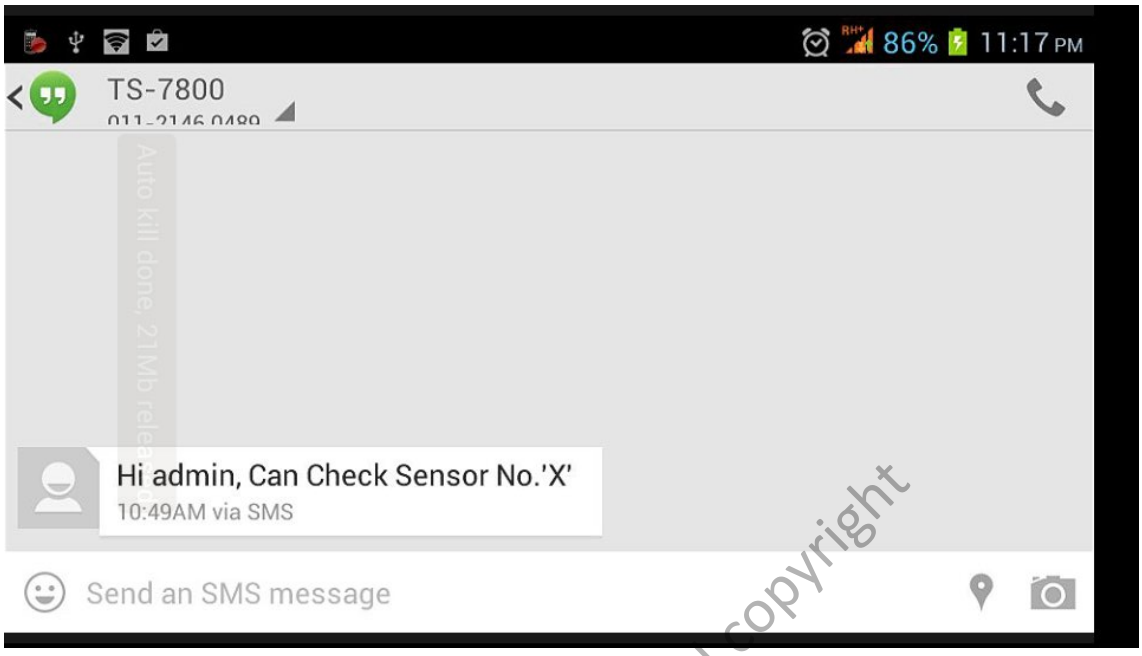


Figure 5.2: First notification by SMS

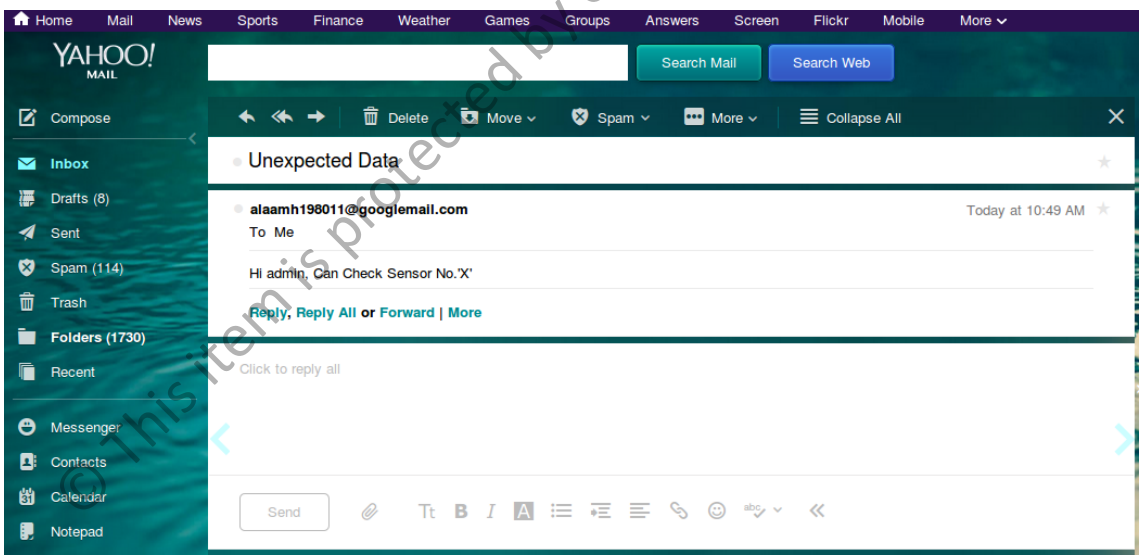
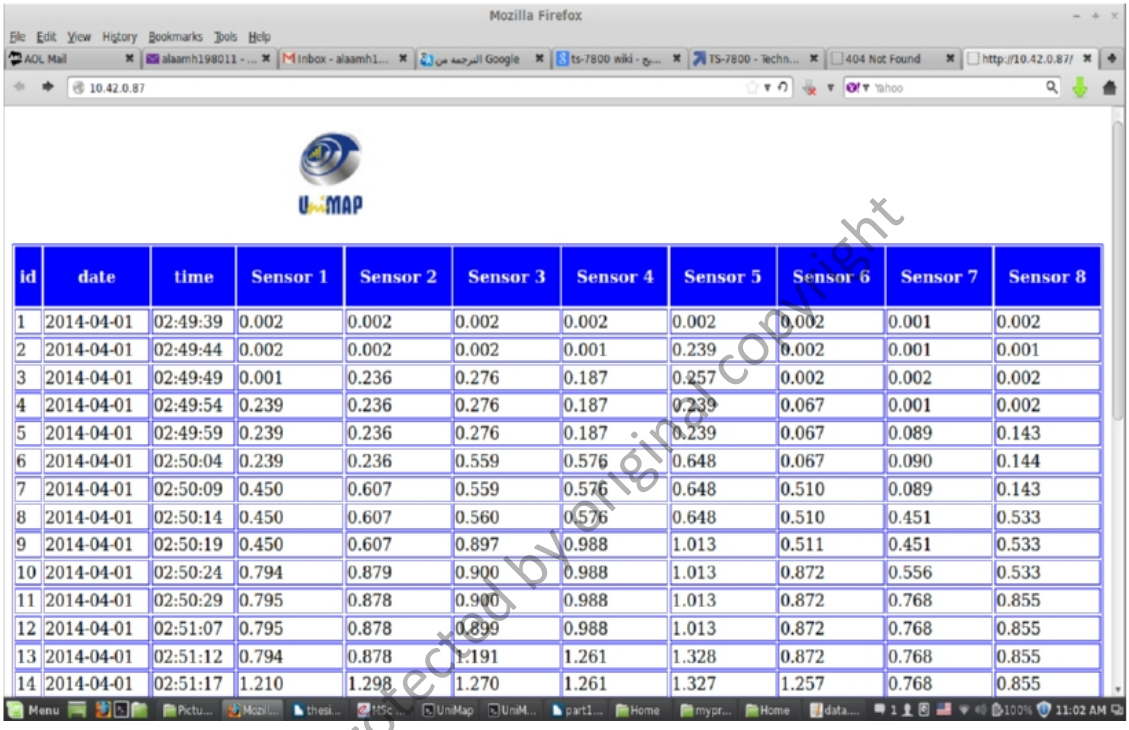


Figure 5.3: Second notification by email

Monitoring Module combines multi-techniques like PHP, HTML, AJAX and Apache2 server to access MySql database through network by web browser. Figure 5.4 shows the main window of monitoring module with eight columns ADC channels and three columns for details.



id	date	time	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Sensor 6	Sensor 7	Sensor 8
1	2014-04-01	02:49:39	0.002	0.002	0.002	0.002	0.002	0.002	0.001	0.002
2	2014-04-01	02:49:44	0.002	0.002	0.002	0.001	0.239	0.002	0.001	0.001
3	2014-04-01	02:49:49	0.001	0.236	0.276	0.187	0.257	0.002	0.002	0.002
4	2014-04-01	02:49:54	0.239	0.236	0.276	0.187	0.239	0.067	0.001	0.002
5	2014-04-01	02:49:59	0.239	0.236	0.276	0.187	0.239	0.067	0.089	0.143
6	2014-04-01	02:50:04	0.239	0.236	0.559	0.576	0.648	0.067	0.090	0.144
7	2014-04-01	02:50:09	0.450	0.607	0.559	0.576	0.648	0.510	0.089	0.143
8	2014-04-01	02:50:14	0.450	0.607	0.560	0.576	0.648	0.510	0.451	0.533
9	2014-04-01	02:50:19	0.450	0.607	0.897	0.988	1.013	0.511	0.451	0.533
10	2014-04-01	02:50:24	0.794	0.879	0.900	0.988	1.013	0.872	0.556	0.533
11	2014-04-01	02:50:29	0.795	0.878	0.900	0.988	1.013	0.872	0.768	0.855
12	2014-04-01	02:51:07	0.795	0.878	0.899	0.988	1.013	0.872	0.768	0.855
13	2014-04-01	02:51:12	0.794	0.878	1.191	1.261	1.328	0.872	0.768	0.855
14	2014-04-01	02:51:17	1.210	1.298	1.270	1.261	1.327	1.257	0.768	0.855

Figure 5.4: Main window of monitoring module

5.2 System Performance

The system performance of the SBC refers to the resources CPU and RAM usage in a certain period of time, ordinarily expressed by the percentage of usable resource to the entire ability. The most general usage of utilization proportion is to find out potential blocking area or bottleneck. The overflow usage of the resources may bring down the SBC. In addition, the measurements of resource utilization rate make us able to identify resources that have low utilization. Table 5.1 shows the average CPU and memory usage before and after system modules execution. In this table indicates user time including nice time, where the time indicates for running non-kernel code , sy called system time which signs the time spent for running kernel code, id refers the CPU idle time and lastly

the wa signs for waiting time for I/O operation.

Table 5.1: Resource Utilization

Version	Average CPU usage (Before System Running)				Average CPU usage (During System Running)				Free Space RAM (Before System Running)	Free Space RAM (During System Running)
	us	sy	id	wa	us	sy	id	wa		
Debian 7 Wheezy	5	5	90	0	4	6	90	0	62816 KB	73196KB

The *top* program equips a dynamic real-time show of a system implementation. It can display system synopsis information in addition to a list of tasks presently being managed by the Linux kernel. This program in Linux can use to measure CPU and Memory load of the SBC for each process.

5.2.1 CPU Utilization

CPU utilization refers to the computer's usage of processing resources, or the amount of work handled by a CPU. Actual CPU utilization varies depending on the amount and type of managed computing tasks. Certain tasks require heavy CPU time, while others require less because of non-CPU resource requirements. The CPU utilization data items CPU idle, CPU I/O wait, CPU nice-level usage, CPU system-level usage and CPU user-level usage are available for monitoring by using top command. When collect all these items the result must be 100% in other word to measure CPU the usage can subtract idle from one hundred for specific time. Figure 5.5 shows the CPU utilization rate graph for TS-7800 SBC.

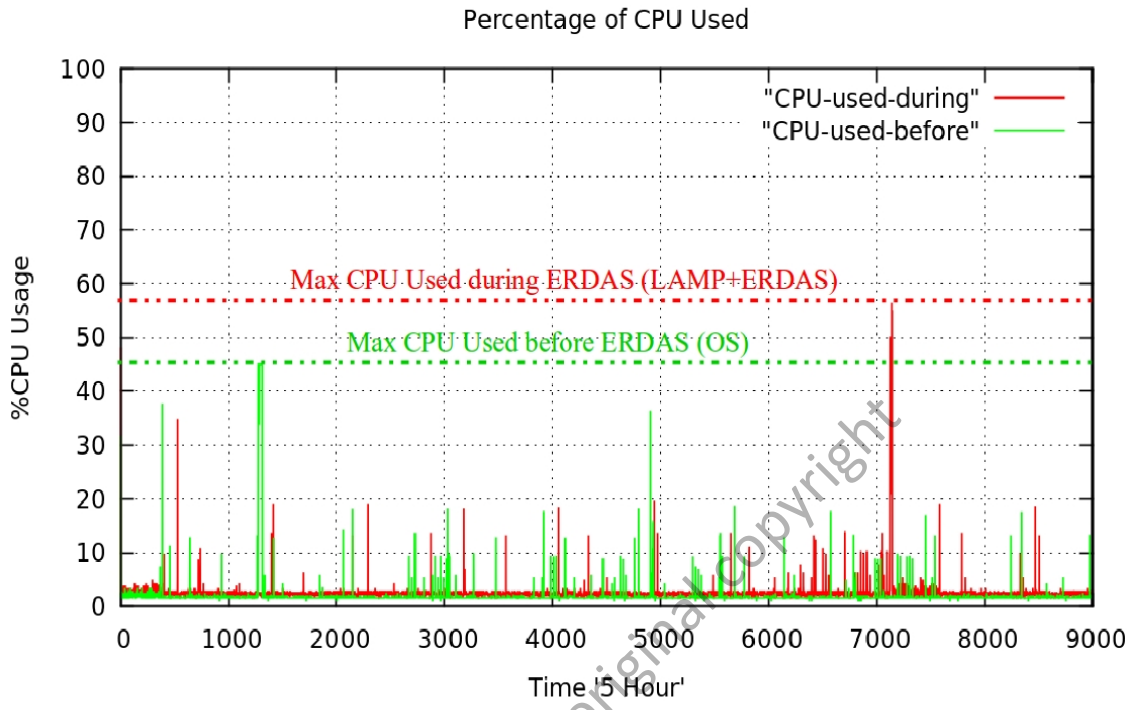


Figure 5.5: CPU Usage (Before and During System Running)

5.2.2 Memory Utilization

Physical memory is a limited resource on any system. The memory handler in the Linux system manages the allocation of that finite resource by freeing sections of physical memory when possible. The TS-7800 memory is 128MB the average used memory before and during running system illustrated in Figure 5.6.

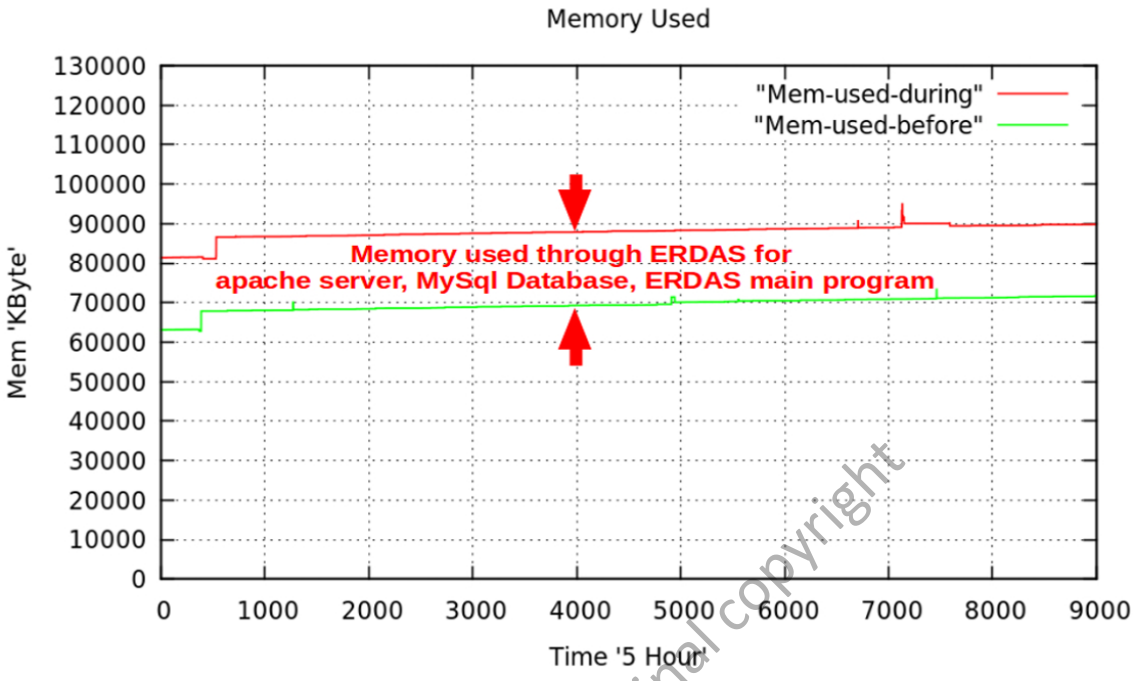


Figure 5.6: Memory Usage (Before and During System Running)

5.3 Summary

This chapter display the results for all modules as screen shots. The initialize module result shows kernel version, modem connected to network, enable PC104 and login immediately. Analysis Module check and analyze all coming data from channels if received unexpected value it will gives command to Notification Module to send SMS and email to user, results of these two modules illustrated in SMS and email screen shot pictures. The last result is for monitoring module display the data collected and saved in database through web browser. The other part of this chapter is system performance to check resource utilization if system running for long time.

CHAPTER 6

CONCLUSION AND RECOMMENDATION

6.1 Conclusion

The goal of the project is to design and implement an system that would fit the needs of the collected data environment whatever the number, type, quantity or quality. The hardware and software architecture both should be chosen depending on sensor equipment demands from minimum to maximum value measurement.

The base of the system that selected the single-board computer ARM9 TS-7800 has been to form database and process unit. All peripherals needed to interface can connected through TS-7800 I/O ports like TS-9700 ADC peripheral board connect to PC104 with adjusted to eight units to get 64 input analog channels and 3G modem connect to USB. Number of peripherals that can be connected to TS-7800 determined by the device drivers in the kernel from manufacturer and to add any new peripheral device should be added own device driver to the kernel and compile to get new kernel supported this device. Detailed information to insert device driver and compile new kernel is given.

To obtain a higher performance and efficiency of the hardware utilization that is guaranteed by using a product supplied custom GNU/Linux operating system because on the top of the GNU/Linux operating system many user space libraries have been implemented to clarify and simplify the control system development. Detailed information on the software principles is given.

Communication amongst the software modules of the system is handled by the simple C programs and Shell scripts written as needed to perform the functions required. The collected data can display through the web browser from any computer tethered to the same Service Set Identification (SSID) network, and if we wanted more broadly can buy and use Domain Name System (DNS). SMS sent to Admin when the system receipt unexpected data is repeated every 15 minutes if the condition persists.

Finally can say successfully developed data acquisition system by using SBC consists

of:

- initialize immediately when connect to power supply without need to configurations.
- Analyze and save collected data in MySql databases on TS-7800 and used it as web server.
- Send notification to one user or more when receive unexpected data from any ADC channel.
- Monitor the collected data through netwoke by web browser.

6.2 Future Work

Can develop this research to create a robot which can collect data, interpret and process it. All this depends on the use sensors, converting analog signal to the digitized and stored to know the robot what around him from the conditions and environment in either case response robot to these data shall be by researching the database, find the required data and take it out through sound or kinetic energy. This work requires the greatest possible number of sensors and ADC to collect data and can take advantage of TS-7800 and TS -9700. Adding to another SBC top-class works in parallel with TS-7800 and connects to the same database regard to the role of voice commands, translate it and answered to give the highest processing speed and less time.

REFERENCES

APC-1. (2014). *APC - Rock.* Retrieved August 1, 2014, from http://apc.io/products/rock/

ARC. (2012). SENSORS Types and Characteristics.

Bakiri, M., Titri, S., Izeboudjen, N., Abid, F., Louiz, F., & Lazib, D. (2012, March). Embedded system with Linux Kernel based on OpenRISC 1200-V3. *2012 6th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT)*, 177–182. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6481909 doi: 10.1109/SETIT.2012.6481909

BeagleBoard.org. (2014). *BeagleBoard.* Retrieved August 1, 2014, from http://beagleboard.org

Berger, A. S. (2002). *Embedded Systems Design: An Introduction to Processes, Tools, and Techniques* (Vol. 2002).

Cooper, M. (2012). Advanced Bash-Scripting Guide An in-depth exploration of the art of shell scripting Table of Contents.

Cubieboard. (2014). *Cubieboard | A series of open source hardware.* Retrieved August 1, 2014, from http://cubieboard.org/

DATAQ Instruments. (2012). DI-145 USB Data Acquisition Starter Kit Analog Inputs protected to DI-145 Close-up.

DATAQ Instruments. (2014). *DATAQ Instruments.* Retrieved August 1, 2014, from http://www.dataq.com/data-acquisition/

Dean, A. G., Conrad, J. M., & Road, W. (2012). *Embedded Systems*.

Debian.org. (2014a). *Debian – Packages.* Retrieved August 1, 2014, from https://www.debian.org/distrib/packages

Debian.org. (2014b). *Debian – The Universal Operating System.* Retrieved August 1, 2014, from https://www.debian.org/

European Community (Information Society, & Technology). (2009). A KM appliance - for deployment in an agricultural knowledge management setting. , 1–29.

Feynman, R. (2007). *Programming Embedded Systems , Second Edition with C and GNU Development Tools*. O'Reilly.

Free Software Foundation. (2014). *GCC, the GNU Compiler Collection - GNU Project - Free Software Foundation (FSF).* Retrieved August 1, 2014, from http://gcc.gnu.org/

Gite, V. (2010). *Linux Shell Scripting Tutorial* (No. May).

Greg Kroah-Hartman, Jonathan Corbet, A. M. (2008). Linux Kernel Development. (March).

Haibo, L., Lina, W., & Chuijie, Y. (2010, June). An Embedded Corrosion Data Acquisition System for Marine Platform. *2010 International Conference on Electrical and Control Engineering*, 380–383. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5629886 doi: 10.1109/iCECE.2010.99

Hallinan, C. (2006). *Embedded Linux Primer: A Practical Real-World Approach*.

Halupka, D. (2002). *ANALYSIS OF SAMPLE AND HOLD CIRCUITS FOR ANALOG TO DIGITAL C ONVERTERS*.

Hardkernel Co. (2013). *ODROID | Hardkernel.* Retrieved August 1, 2014, from http://www.hardkernel.com/main/main.php

Huawei Technologies Co. (2013). Guide to Kernel Driver Integration in Android for Huawei Modules. , 1–26.

Jaan. (2013). *TS-7000 ARM SBC - Yahoo Groups.* Retrieved August 1, 2014, from https://groups.yahoo.com/neo/groups/ts-7000/conversations/topics/22375

John Park, S. M. (2003). *Practical Data Acquisition for Instrumentation and Control Systems*.

Li, X.-h., & Liu, J.-h. (2010). Design of Embedded Data Acquisition and Remote Control System Based on Linux. *IEEE 2010*, *3*(Iccasm), 299–302.

Linux.org. (2014). *Introduction to Linux, Free Software and Open Source.* Retrieved August 1, 2014, from http://linux.org.au/introduction-linux-free-software-and-open-source

Madisetti, E. V. K., Williams, D. B., Kosonocky, S., & Xiao, P. (1999). Kosonocky , S . & Xiao, P. " Analog-to-Digital Conversion Architectures ".

MC Corporation. (2012). *Data Acquisition*.

MC Corporation. (2014). User's Guide. (January).

Mygind, L., Jacobsen, R. H., & Swirtun, O. (2006). Introducing Linux and open source. (1), 30–35.

National Instruments. (2104). *Data Acquisition (DAQ) - National Instruments.* Retrieved August 1, 2014, from http://www.ni.com/data-acquisition

Olimex Ltd. (2014). *A13 - Open Source Hardware Boards.* Retrieved August 1, 2014, from https://www.olimex.com/Products/OLinuXino/A13/

Peng, D., Zhang, H., Zhang, K., Li, H., & Xia, F. (2009, February). Research and Development of the Remote I/O Data Acquisition System Based on Embedded ARM Platform. *2009 International Conference on Electronic Computer Technology*, 341–344. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4795979 doi: 10.1109/ICECT.2009.26

Pico Technology Ltd. (2013). High-Resolution Data Loggers.

QuickEmbed Technology Co. (2014). *Hackberry A10 Dev Board.* Retrieved August 1, 2014, from http://www.quickembed.com/Tools/Shop/A8/201301/266.html

Raspberry Pi Foundation. (2014). *Raspberry Pi.* Retrieved August 1, 2014, from http://www.raspberrypi.org/

Scholar, P. G. (2011). Design of On-line Interactive Data Acquisition and Control System for Embedded Real Time Applications. , 551–556.

SECO USA Inc. (2014). *UDOO: Android Linux Arduino in a tiny single-board computer.* Retrieved August 1, 2014, from http://www.udoo.org/

Simmonds, C. (2010). The Embedded Linux Quick Start Guide Kernel and user space.

Technologic Systems Inc. (2009). TS-9700 Manual. , 1–11.

Tiwari, A. (2012). A low power high speed dual data rate acquisition system using FPGA. *2012 International Conference on Communication, Information & Computing Technology (ICCICT)*, 1–4. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6398101 doi: 10.1109/ICCICT.2012.6398101

APPENDICES

Appendix A: Linux Commands

Following are some commonly used GNU/Linux command for this project.

Command	Description
$apt-get$	Advanced Package Tool handling utility – command-line interface.
cat	Display the contents of a file (concatenate).
cd	Change the current working directory to a specific Folder.
$chat$	Automated conversational script with a modem.
$chmod$	Change file and directory access permissions.
$chroot$	Run a command with a different root directory.
$date$	Display or change the date and time.
dd	Dump Data - convert and copy a file (use for RAW storage).
df	Display free disk space. With no arguments, df reports the space used and available on all currently mounted filesystems (of all types). Otherwise, df reports on the filesystem containing each argument file.
$echo$	Display message on screen, writes each given STRING to standard output, with a space between each and a newline after the last one.
$fdisk$	Manipulate disk partition table.
$find$	Search a folder hierarchy for filename(s) that meet a desired criteria.
$free$	Display amount of free and used memory in the system.
gcc	GNU project C and C++ compiler.
$gedit$	Text editor for the GNOME Desktop.
$getty$	Opens a tty port, prompts for a login name and invokes the /bin/login command.
$gparted$	Gnome partition editor for manipulating disk partitions.
$init$	Process management daemon.

$kill$	Stop a process from running, either via a signal or forced termination.
ln	Make links between files, by default, it makes hard links; with the $-s$ option, it makes symbolic (or "soft") links.
ls	List information about FILEs, by default the current directory.
$make$	GNU make utility to maintain groups of programs.
man	Display helpful information about commands.
$mailx$	Send and receive Internet mail.
$mkdir$	Create new folder(s), if they do not already exist.
$mkfs.vfat$	Create an MS-DOS file system under Linux.
$mount$	Mount a file system. All files accessible in a UNIX system are arranged in one big tree, the file hierarchy, rooted at /. These files can be spread out over several devices. The mount command serves to attach the file system found on some device to the big file tree.
mv	Move or rename files or directories.
$nano$	Small, free and friendly editor, the default editor included in the non-free Pine package.
$passwd$	Modify a user password.
$pkill$	Kill all processes matching the search text.
ps	Process status, information about processes running in memory. If want a repetitive update of this status, use top.
pwd	Print Working Directory.
rm	Remove files (delete/unlink).
$rmdir$	Remove folder(s), if they are empty.
$system$	Execute a shell command.
su	Substitute user identity. Run a command with substitute user and group id, allow one user to temporarily become another user. It runs a command (often an interactive shell) with the real and effective user id, group id, and supplemental groups of a given USER.
tar	Tape Archiver, store, list or extract files in an archive.

top	Provide information (frequently refreshed) about the most CPU-intensive processes currently running. See ps for explanations of the field descriptors.
$tune2fs$	Adjust tunable filesystem parameters on ext2/ext3/ext4 filesystems
$vmstat$	Report virtual memory statistics.
$wget$	The non-interactive network downloader.

Appendix B: C programs code

B.1 DAS.c to read analog signals, ADC, Write Data to MySql table With Date & Time, checking data level and send notificaion.

```
/*** To Create MySql Table for this Program The Command is ***/
/*
create table table0 (
id INT UNSIGNED NOT NULL AUTO_INCREMENT,
PRIMARY KEY (id),
date char(40),
time char(40),
Ch0 char(20),
Ch1 char(20),
Ch2 char(20),
Ch3 char(20),
Ch4 char(20),
Ch5 char(20),
Ch6 char(20),
Ch7 char(20) );
*/
//To Compile
//   $gcc DAS.c -o DAS `mysql_config --cflags --libs`

#include <stdio.h>
#include <fcntl.h>
#include <sys/mman.h>

#include <my_global.h>
#include <mysql.h>
#include <string.h>
```

```
#define BASE 0xEE000000
#define ADBASE 0x160
#define ADCOMMAND ADBASE
#define ADDATA ADBASE + 2
#define ADREADYBIT 7
#define NUM_CHANNELS 8

typedef unsigned char byte;

int main(void)
{
byte *base;
volatile byte *adcommand;
volatile byte *lsb;
volatile byte *msb;
volatile byte *ldac;
volatile byte *hdac;
unsigned short int val;

char buffer[1024];
char buf0[60];
char buf1[60];
char buf2[60];
char buf3[60];
char buf4[60];
char buf5[60];
char buf6[60];
char buf7[60];
int fd;
float volts;
int i,j;
```

```
fd = open ("/dev/mem" , O_RDWR|O_SYNC);
base = (byte *)mmap(0, getpagesize(), PROT_READ
| PROT_WRITE, MAP_SHARED, fd, BASE);
adcommand = base + ADCOMMAND;
lsb = base + ADDATA;
msb = base + ADDATA + 1;
// start while loop
while (1) {
//Time Structure
struct tm *local;
time_t t,curtime;
char buf_Date[50];
char buf_Time[50];
strcpy(buf_Date, "");  //Clear Data Buffer
strcpy(buf_Time, "");  //Clear Time Buffer
curtime=time(NULL);
strftime(buf_Date, 40, "%Y-%m-%d", localtime(&curtime));
strftime(buf_Time, 40, "%H:%M:%S", localtime(&curtime));
for (j=0; j<NUM_CHANNELS; j++)
{
*adcommand = j;
while(!(*adcommand & (1 << ADREADYBIT)));
volts = (256 * *msb + *lsb) * 2.5 / 4096;
// check sensors
if (volts <= 2.499)
      {
         printf ("Sensors Connected && DAS Running Immediately");
      }
   else
      {
         printf ("Sensors not Connected") && system ("alert");
```

```
        }
//start Switch
switch (j)
{
case (0):
if (volts>2.48)       sprintf(buf0, "%1.3fERR0   ", volts)
&& system ("sms-mail");
//else if (volts<=0) sprintf(buf0, "%1.3fNSen0 ", volts);
else                  sprintf(buf0, "%1.3f       ", volts);
break;

case (1):
if (volts>2.48)       sprintf(buf1, "%1.3fERR1   ", volts)
&& system ("sms-mail");
//else if (volts<=0) sprintf(buf1, "%1.3fNSen1 ", volts);
else                  sprintf(buf1, "%1.3f       ", volts);
break;

case 2:
if (volts>2.48)       sprintf(buf2, "%1.3fERR2   ", volts)
&& system ("sms-mail");
//else if (volts<=0) sprintf(buf2, "%1.3fNSen2 ", volts);
else                  sprintf(buf2, "%1.3f       ", volts);
break;

case 3:
if (volts>2.48)       sprintf(buf3, "%1.3fERR3   ", volts)
&& system ("sms-mail");
//else if (volts<=0) sprintf(buf3, "%1.3fNSen3 ", volts);
else                  sprintf(buf3, "%1.3f       ", volts);
break;
```

```
case 4:
if (volts>2.48)       sprintf(buf4, "%1.3fERR4  ", volts)
&& system ("sms-mail");
//else if (volts<=0) sprintf(buf4, "%1.3fNSen4 ", volts);
else                  sprintf(buf4, "%1.3f      ", volts);
break;

case 5:
if (volts>2.48)       sprintf(buf5, "%1.3fERR5  ", volts)
&& system ("sms-mail");
//else if (volts<=0) sprintf(buf5, "%1.3fNSen5 ", volts);
else                  sprintf(buf5, "%1.3f      ", volts);
break;

case 6:
if (volts>2.48)       sprintf(buf6, "%1.3fERR6  ", volts)
&& system ("sms-mail");
//else if (volts<=0) sprintf(buf6, "%1.3fNSen6 ", volts);
else                  sprintf(buf6, "%1.3f      ", volts);
break;
case 7:
if (volts>2.48)       sprintf(buf7, "%1.3fERR7  ", volts)
&& system ("sms-mail");
//else if (volts<=0) sprintf(buf7, "%1.3fNSen7 ", volts);
else                  sprintf(buf7, "%1.3f      ", volts);
break;
}//End Switch
}//End Loop
//Database Connection
sprintf(buffer ,"INSERT INTO table0 (date, time, Ch0, Ch1, Ch2, Ch3,
```

```
Ch4, Ch5, Ch6, Ch7 )
VALUES ('%s\',\'%s\', '%s', '%s', '%s', '%s', '%s', '%s', '%s', '%s')",
buf_Date, buf_Time, buf0, buf1, buf2, buf3, buf4, buf5, buf6, buf7);
MYSQL *connect;
connect = mysql_init(NULL);
mysql_real_connect(connect,"localhost","root","redhat",
 "mysql", 0, NULL, 0);

if(mysql_query(connect,buffer))
 {
      printf("Error %u: %s\n",mysql_errno(connect),mysql_error(connect));
      exit(1);
  }
mysql_close(connect);
sleep(5);
} //End While
}
```

B.2 This code to control on RDAS by matrix keypad and display the command on LCD, contains the following comannds : a- 0 to shutdown. b- 1 to Run RDAS. c- 2 to Stop RDAS. d- up to connect to network. e- down to disconnect network. f- clear to delete all data. g- Help to display buttons functions.


```
/* the binary output of this program
trnasfer to sbin directory */

/*                cp  ekey /sbin/          */
#include<unistd.h>
#include<sys/types.h>
#include<sys/mman.h>
#include<sys/time.h>
#include<stdio.h>
```

```
#include<fcntl.h>
#include<string.h>
#include <stdlib.h>

#define DIOBASE 0xE8000000
#define PINMASK(x) (1 << (x << 1))

#define KEY_0 2
#define KEY_1 5
#define KEY_2 6
#define KEY_3 7
#define KEY_4 9
#define KEY_5 10
#define KEY_6 11
#define KEY_7 13
#define KEY_8 14
#define KEY_9 15
#define KEY_UP 8
#define KEY_DOWN 12
#define KEY_2ND 16
#define KEY_CLEAR 1
#define KEY_HELP 3
#define KEY_ENTER 4


int printkey(unsigned int key);
unsigned int getkey(volatile unsigned int *dioptr);
int accept_keypress(volatile unsigned int *dioptr);

int main() {
  unsigned int key;
```

```
  volatile unsigned int *dioptr;
  int fd = open("/dev/mem", O_RDWR|O_SYNC);

  dioptr = (unsigned int *)mmap(0, getpagesize(),
    PROT_READ|PROT_WRITE, MAP_SHARED, fd, DIOBASE);

  printf("Press some keys, or enter to quit:\n");
  do {
    key = (accept_keypress(dioptr));
    printkey(key);
    } while (key != KEY_ENTER);

  return 0;
  }


/* accept_keypress does not return
until a key is pressed and released. */

int accept_keypress(volatile unsigned int *dioptr) {
  unsigned int ret = 0, counter = 0;

  while (ret == 0) {
/*    usleep(10); */
    ret = getkey(dioptr);
    }
  do {
    counter++;
    if (getkey(dioptr) == ret) {
      counter = 0;
      }
```

```
    usleep(1);
    } while (counter < 10);
  return ret;
  }

/* getkey checks the inputs once and returns
0 for no key or 1 to 16 for a keypress */
unsigned int getkey(volatile unsigned int *dioptr) {

  unsigned int i1, i2, response, ret = 0;

  for(i1 = 0; i1 < 4; i1++) {

/* write to all pins but one */
    *(dioptr + 0x08/sizeof(unsigned int)) = 0x5555u ^ PINMASK(i1);

/* filter out zeros in output */
    response = 0x5500u ^ (0x5500u &
    *(dioptr + 0x04/sizeof(unsigned int)));

/* response is zero if no keys on the current row */
    if (response) {
      for(i2 = 0; i2 < 4; i2++) {
        if(PINMASK(i2) & (response >> 8)) { ret = 4 * i1 + i2 + 1;}
        }
      }
    }
  return ret;
  }

int printkey(unsigned int key) {
```

```
switch (key) {
  case KEY_CLEAR:
       system ("truncate-t");
       system ("tolcd all_Data_deleted");
       sleep (3);
       system ("tolcd");
    break;
  case KEY_0:
       system ("tolcd RDAS_will_go_to_Shutdown after_[5]_sec");
       sleep (1);
       system ("tolcd RDAS_will_go_to_Shutdown after_[4]_sec");
       sleep (1);
       system ("tolcd RDAS_will_go_to_Shutdown after_[3]_sec");
       sleep (1);
       system ("tolcd RDAS_will_go_to_Shutdown after_[2]_sec");
       sleep (1);
       system ("tolcd RDAS_will_go_to_Shutdown after_[1]_sec");
       sleep (1);
       system ("tolcd RDAS_will_go_to_Shutdown after_[0]_sec");
       sleep (1);
       system ("init 0");
    break;
  case KEY_HELP:
      system ("tolcd 0-shutdown_1-Run-RDAS 2-Stop-RDAS");
      sleep (3);
      system ("tolcd up-connect_down-disconnect clear-delete-data");
      sleep (3);
      system ("tolcd 0-shutdown_1-Run-RDAS 2-Stop-RDAS");
      sleep (3);
      system ("tolcd up-connect_down-disconnect clear-delete-data");
      sleep (3);
```

```
    system ("tolcd");
  break;
case KEY_ENTER:
  printf("enter");
  break;
case KEY_1:
    system ("RDAS&");
    system ("tolcd RDAS-turn-ON");
    sleep (3);
    system ("tolcd");
  break;
case KEY_2:
    system ("pkill RDAS");
    system ("tolcd RDAS-turn-OFF");
    sleep (3);
    system ("tolcd");
  break;
case KEY_3:
  printf("3");
  break;
case KEY_UP:
    system ("sakis3g connect&");
    system ("tolcd connect-to-network");
    sleep (3);
    system ("tolcd");
  break;
case KEY_4:
  printf("4");
  break;
case KEY_5:
  printf("5");
```

```
  break;
case KEY_6:
      system ("tolcd RDAS_will_go_to_Restart after_[5]_sec");
      sleep (1);
      system ("tolcd RDAS_will_go_to_Restart after_[4]_sec");
      sleep (1);
      system ("tolcd RDAS_will_go_to_Restart after_[3]_sec");
      sleep (1);
      system ("tolcd RDAS_will_go_to_Restart after_[2]_sec");
      sleep (1);
      system ("tolcd RDAS_will_go_to_Restart after_[1]_sec");
      sleep (1);
      system ("tolcd RDAS_will_go_to_Restart after_[0]_sec");
      sleep (1);
      system ("init 6");
  break;
case KEY_DOWN:
      system ("sakis3g disconnect&");
      system ("tolcd disconnect-to-network");
      sleep (3);
      system ("tolcd");
  break;
case KEY_7:
  printf("7");
  break;
case KEY_8:
  printf("8");
  break;
case KEY_9:
  printf("9");
  break;
```

```
    case KEY_2ND:
      system ("edate");
      break;
    default:
      printf("nothing");
      break;
    }
  printf("\n");
  return 0;
  }
```

B.3 Other program used with matrix keypad to inmplement functios through it. truncute-t.c is a C code to delete all date from database table.


```
/* gcc truncate-t.c -o truncate-t `mysql_config --cflags âĂŞlibs` */
/*        cp  truncate-t /sbin/    */
#include <my_global.h>
#include <mysql.h>

void finish_with_error(MYSQL *con)
{
  fprintf(stderr, "%s\n", mysql_error(con));
  mysql_close(con);
  exit(1);
}
int main(int argc, char **argv)
{
  MYSQL *con = mysql_init(NULL);

  if (con == NULL)
  {
      fprintf(stderr, "%s\n", mysql_error(con));
```

```
    exit(1);
}

if (mysql_real_connect(con, "localhost", "root",
"redhat", "mysql", 0, NULL, 0) == NULL)
{
    finish_with_error(con);
}

if (mysql_query(con, "TRUNCATE TABLE tableA;")) {
    finish_with_error(con);
}
mysql_close(con);
exit(0);
```

Appendix C: Web source code

C.1 AJAX code to reload data from MySql each 1 second.

```
var seconds = 1;
var divid = "timediv";
var url = "boo.php";

function refreshdiv(){

var xmlHttp;
try{
xmlHttp=new XMLHttpRequest(); // Firefox, Opera 8.0+, Safari
}
catch (e){
try{
xmlHttp=new ActiveXObject("Msxml2.XMLHTTP"); // Internet Explorer
}
catch (e){
try{
xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
}
catch (e){
alert("Your browser does not support AJAX.");
return false;
}
}
}

fetch_unix_timestamp = function()
{
return parseInt(new Date().getTime().toString().substring(0, 10))
}
```

```
var timestamp = fetch_unix_timestamp();
var nocacheurl = url+"?t="+timestamp;

// The code...

xmlHttp.onreadystatechange=function(){
if(xmlHttp.readyState==4){
document.getElementById(divid).innerHTML=xmlHttp.responseText;
setTimeout('refreshdiv()',seconds*1000);
}
}
xmlHttp.open("GET",nocacheurl,true);
xmlHttp.send(null);
}

// Start the refreshing process

var seconds;
window.onload = function startrefresh(){
setTimeout('refreshdiv()',seconds*1000);
}
```


C.2 PHP code to access Database through web browser.

```
<?php
$host="localhost"; // Host name
$username="root"; // Mysql username
$password="redhat"; // Mysql password
$db_name="mysql"; // Database name
$tbl_name="table0"; // Table name
$conn=mysql_connect("$host", "$username", "$password")or die("cannot connect
mysql_select_db("$db_name")or die("cannot select DB");
```

```
$sql="select * from ".$tbl_name.";";
$result = mysql_query($sql) or die(mysql_error());
?>
<head>
<style>
table ,td ,th
        {
         border:1px solid blue;
        }
th
        {
         background-color:blue;
         color:white;
        }

table
        {
         width:100%;
        }
th
        {
         height:50px;
        }
</style>
</head>
<table border="3" width="500" border="0" align="center">
     <tr class="odd">

        <th>id</th>
        <th>date</th>
        <th>time</th>
```

```
            <th>Sensor 1</th>
            <th>Sensor 2</th>
            <th>Sensor 3</th>
            <th>Sensor 4</th>
            <th>Sensor 5</th>
            <th>Sensor 6</th>
            <th>Sensor 7</th>
            <th>Sensor 8</th>
        </tr>
        <?php while ($row = mysql_fetch_array($result)) : ?>
        <tr>
<td>
<?php echo $row["id"]; ?>
</td>
<td>
<?php echo $row["date"]; ?>
</td>
<td>
<?php echo $row["time"]; ?>
</td>
<td>
<?php echo $row["col0"]; ?>
<td>
<?php echo $row["col1"]; ?>
</td>
<td>
<?php echo $row["col2"]; ?>
</td>
<td>
<?php echo $row["col3"]; ?>
</td>
```

```
<td>
<?php echo $row["col4"]; ?>
</td>
<td>
<?php echo $row["col5"]; ?>
</td>
<td>
<?php echo $row["col6"]; ?>
</td>
<td>
<?php echo $row["col7"]; ?>
</td>
    </tr>
    <?php endwhile; ?>
</table>
```

C.3 HTML main webpage code.

```
<!DOCTYPE html>
<html>
  <img src="1.png" alt="logo" />
<script src="ajax.js"></script>

<strong> </strong>

<script type="text/javascript"><!--
refreshdiv();
// --></script>
<div id="timediv"></div>

</html>
```