Spring 2-5-2009

M4 Controller Diagnostic Database

Naveen Kumar Kota
Dakota State University

M4 Controller Diagnostic Database

A graduate project submitted to Dakota State University in partial fulfillment of the requirements for the degree of

Master of Science

In

Information Systems

5 Feb, 2009

By

Naveen Kumar Kota

Project Committee:

Dr. Ronghua Shan

Dr. Stephen Krebsbach

Dr. Joyce Havlik

DAKOTA STATE
dsu
UNIVERSITY

PROJECT APPROVAL FORM

We certify that we have read this project and that, in our opinion, it is satisfactory in scope and quality as a project for the degree of Master of Science in Information Systems.

Student Name: Naveen Kumar Kota

Master's Project Title: M4 Controller Diagnostic Database

Faculty supervisor: Ronghua Shan Date: 3-30-2009

Committee member: Joyce Havlik Date: 3-31-2009

Committee member: Stephen Krebsbach Date: 3-31-09

ACKNOWLEDGMENT

I want to thank Daktronics for giving an opportunity to work on this Project as Student Temporary Position; My supervisors Arthur Jelsma, Steve Springman very much for their directions on this project; My Colleagues Steve Bostrom, Jeremy Vanorny, Brad Borden, Kevin Pond and Barbara Fourney for many helpful suggestions on the development.

I also thank Dr. Ronghua Shan, Dr. Stephen krebsbach, Dr. Joyce Havlik for their precious time and discussions about the project.

Last but not the least, I cordially thank Dr.Omar El-Gayar, my Father, my Mother, my Sister, roommates Shyam Kumar Minda Shankar, Suman, Vineel and Friends for their consistent encouragement for my endeavor to pursue the Master of Science degree in Information Systems.

ABSTRACT

The M4 controller which is under development is the latest version for the M2 and M3 based Controller. The purpose of the controller design is to provide a core platform that can be used across multiple display products. Currently M2 and M3 Controllers retrieve the data from MLC and Modules' and transfer data to the Server side Diagnostic Database which generate notifications using analysis engine. In the M4 controller, this notification is generated from the controller instead of the server side by using the data which are available at the user level database. Currently, the primary goal of the M4 controller is to build a diagnostic database at the user level to generate the notifications.

M4 uses Embedded Linux based platform using Ubuntu Operating System. Research it found that SQLite is a self-contained, server less, Zero configuration and transactional Database Engine that to build the user level Database on the M4 Controller by retrieving the data from Dynamic, Static, Various and Created type files from kernel level diagnostic database using API.

Diagnostics - The process of recording operational status of display hardware components.

Diagnostic Database- is the storage of all the operational status of display Hardware components and notifications which need to be generated.

DECLARATION

I hereby certify that this project constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions or writings of another.

I declare that the project describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,

_Naveen Kumar Kota__

<Student name>

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The M4 controller which is under the development is going to be a latest version for M2 and M3 based Controller. The purpose of the controller design is to provide a core platform that can be used across multiple display products. Currently M2 and M3 Controllers retrieve the data from Multiple Line Controller (MLC) and Modules' and transfer data to the Server side Diagnostic Database which generate notifications using analysis engine, in the M4 controller this notification should be generated from controller instead of server side by using the data which is available in controller at user level Database. Now the primary goal of the M4 controller is to build a diagnostic database at the user level to generate the notifications.

M4 uses an embedded Linux based platform using the Ubuntu Operating System. By research it found that SQLite is the self-contained, server less, Zero configuration and transactional Database Engine which would be helpful to build the user level Database on the M4 Controller by retrieving the data from dynamic, static, various and created type files from a kernel level diagnostic database using API.

Diagnostics is the process of recording the operational status of display hardware components.

Diagnostic database is the storage of all the operational status of display hardware components and notification which are needed to be generated.

CHAPTER 2

LITERATURE REVIEW

2.1 SQLite

2.1.1 SQLite Is Self-Contained

SQLite is largely self-contained. It requires very minimal support from external libraries or from the operating system. This makes it well suited for use in embedded devices that lack the support infrastructure of a desktop computer. This also makes SQLite appropriate for use within applications that need to run without modification on a wide variety of computers of varying configurations. SQLite is written in ANSI-C and should be easily compiled by any standard C compiler. It makes minimal use of the standard C library. The only required C library functions called are:

* memset()

* memcpy()

* memcmp()

* strcmp()

* malloc(), free(), and realloc()

SQLite can be configured at start-time to use a static buffer in place of calling malloc for the memory it needs. The date and time Structured Query Language (SQL) functions provided by SQLite require some additional C library support, but those functions can also be omitted from the build using compile-time options.

Communications between SQLite and the operating system and disk are mediated through an interchangeable Virtual File System (VFS) layer. VFS modules for Unix (linux and MacOSX), OS/2, Win32, and WinCE are provided in the source tree. It is a simple matter to devise an alternative VFS for embedded devices.

For safe operation in multi-threaded environments, SQLite requires the use of mutexes. Appropriate mutex libraries are linked automatically for Win32 and Unix platforms. For other systems, mutex primitives can be added at start-time. Mutexes are only required if SQLite is used by more than one thread at a time.

The SQLite source code is available as an "amalgamation" - a single large C source code file. Projects that want to include SQLite can do so simply by dropping this one source file (named "sqlite3.c") and its corresponding header ("sqlite3.h") into their source tree and compiling it together with the rest of the code. SQLite does not link against any external libraries (other than the C library, as described above) and does not require any special build support.

2.1.2 SQLite Is Serverless

Most SQL database engines are implemented as a separate server process. Programs that want to access the database communicate with the server using some kind of interprocess communication (typically TCP/IP) to send requests to the server and to receive back results. SQLite does not work this way. With SQLite, the process that wants to access the database reads and writes directly from the database files on disk. There is no intermediary server process. There are advantages and disadvantages to being serverless. The main advantage is that there is no separate server process to install, setup, configure, initialize, manage, and troubleshoot. This is one reason why SQLite is a "zero-configuration" database engine.

Programs that use SQLite require no administrative support for setting up the database engine before they are run. Any program that is able to access the disk is able to use a SQLite database. On the other hand, a database engine that uses a server can provide better protection from bugs in the client application - stray pointers in a client cannot corrupt memory on the server. A server is a single persistent process so it is able control database access with more precision, allowing for finer grain locking and better concurrency.

Most SQL database engines are client/server based. Of those that are serverless, SQLite is the only one known to this author that allows multiple applications to access the same database at the same time.

2.1.3 SQLite Is a Zero-Configuration Database

SQLite does not need to be "installed" before it is used. There is no "setup" procedure. There is no server process that needs to be started, stopped, or configured. There is no need for an administrator to create a new database instance or assign access permissions to users. SQLite uses no configuration files. Nothing needs to be done to tell the system that SQLite is running. No actions are required to recover after a system crash or power failure. There is nothing to troubleshoot SQLite just works. Other database engines may run great once you get them going but doing the initial installation and configuration can often be intimidating.

2.1.4 SQLite is Transactional

A transactional database is one in which all changes and queries appear to be Atomic, Consistent, Isolated, and Durable (ACID). SQLite implements serializable transactions that are Atomic, Consistent, Isolated, and Durable, even if the transaction is interrupted by a program crash, an operating system crash, or a power failure to the computer.

We restate and amplify the previous sentence for emphasis: all changes within a single transaction in SQLite either occur completely or not at all, even if the act of writing the change out to the disk is interrupted by

* a program crash,

* an operating system crash, or

* a power failure.

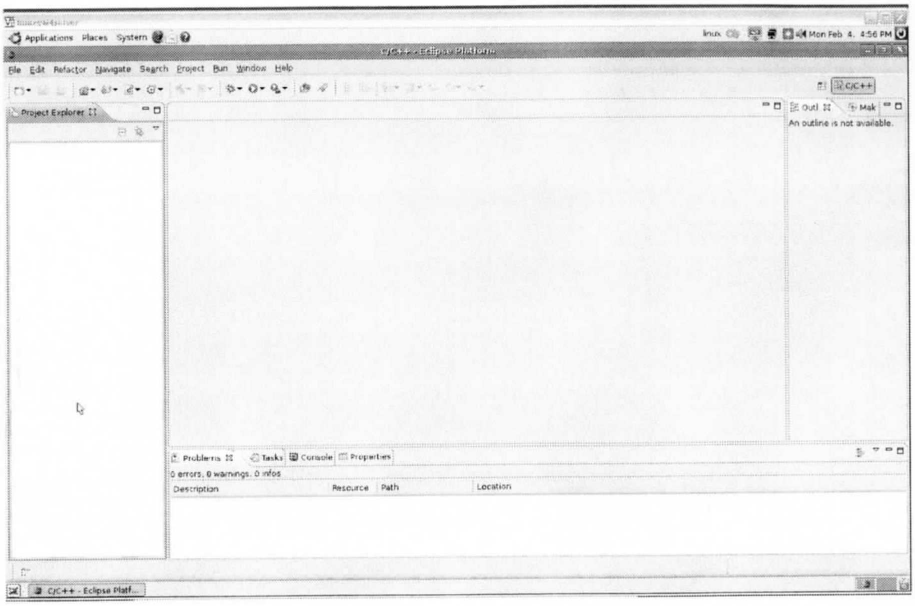
The claim of the previous paragraph is extensively checked in the SQLite regression test suite using a special test harness that simulates the effects on a database file of operating system crashes and power failures.

2.2 M4 Debugging Setup

Tools Required

- PC Running Ubuntu 7.10 Linux or newer (www.ubuntu.com)
- Eclipse IDE CDT C++ (www.eclipse.org)
- JRE 5.0 or newer
- M4 Linux Tools 0.0.3 or newer

After installing the above files, open the Eclipse development environment.



An outline is not available.

0 errors, 0 warnings, 0 infos

Figure 1: Eclipse edit page

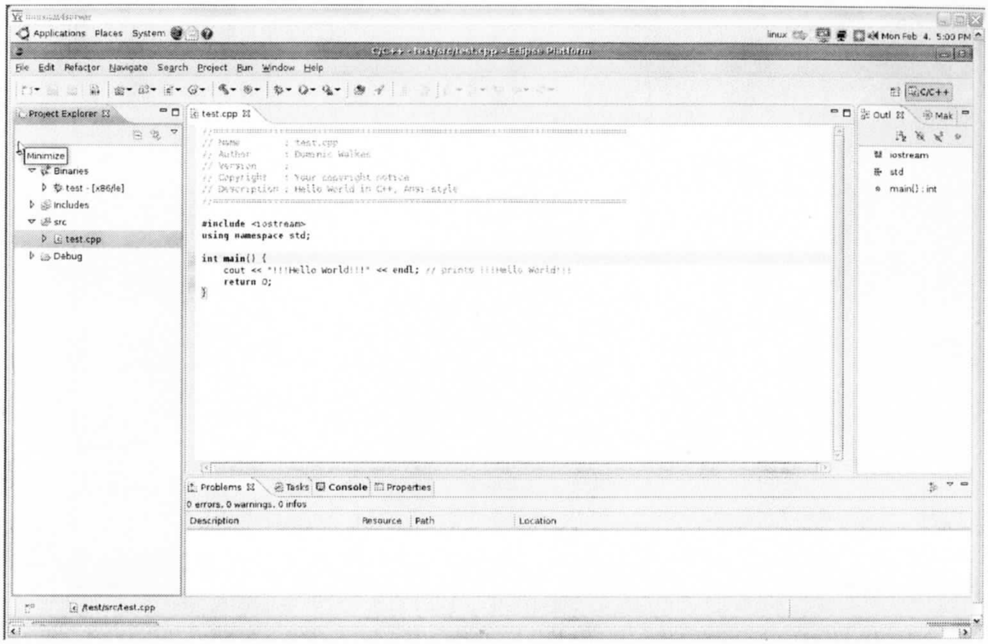
Select File->Project->C++ Project

Project Name: test

Under Executable – select Hello World C++ Project

Toolchain: Linux GCC

Select Finish



```
//============================================================================
// Name        : test.cpp
// Author      : Dominic Walker
// Version     :
// Copyright   : Your copyright notice
// Description : Hello World in C++, Ansi-style
//============================================================================

#include <iostream>
using namespace std;

int main() {
    cout << "!!!Hello World!!!" << endl; // prints !!!Hello World!!!
    return 0;
}
```

0 errors, 0 warnings, 0 infos

Project->Properites

Under" C/C++ Build" Select "Settings"

The Command: g++ is using the internal g++ compiler for the x86, so lets call out the cross compiler for the powerpc here – if you installed your tools to "opt/crosstool" – which is the default, you should be able to link in with:

GCC C++ Compiler: /opt/crosstool/gcc-4.0.2-glibc-2.3.6/powerpc-405-linux-gnu/bin/powerpc-405-linux-gnu-g++

GCC C Compiler: /opt/crosstool/gcc-4.0.2-glibc-2.3.6/powerpc-405-linux-gnu/bin/powerpc-405-linux-gnu-gcc-4.0.2

GCC C++ Linker: /opt/crosstool/gcc-4.0.2-glibc-2.3.6/powerpc-405-linux-gnu/bin/powerpc-405-linux-gnu-g++

GCC Assembler: /opt/crosstool/gcc-4.0.2-glibc-2.3.6/powerpc-405-linux-gnu/bin/powerpc-405-linux-gnu-as

Rebuild the project with the new compiler settings.

Now we are ready to try and run the program. Using a File Transfer Protocol (FTP) program – Firefox FireFTP is handy, connect to a M4/Evaluation board. Transfer the "test" file from your Linux workspace onto the remote target. After transferring, select the file on the remote target and change the properties to allow executing.

Connect over either serial or Transmission Control Protocol (TCP) telnet. Launch the test file with:

"<stored directory>/test" – you should see "hello world" print on the console.

Now we are ready to set up a simple debug session. First, FTP the following file to the working directory that you downloaded "test" into -

/opt/crosstool/PPC405_Debugger/gdbserver/bin/gdbserver

You may need to change the file permissions to allow executing this program as well.

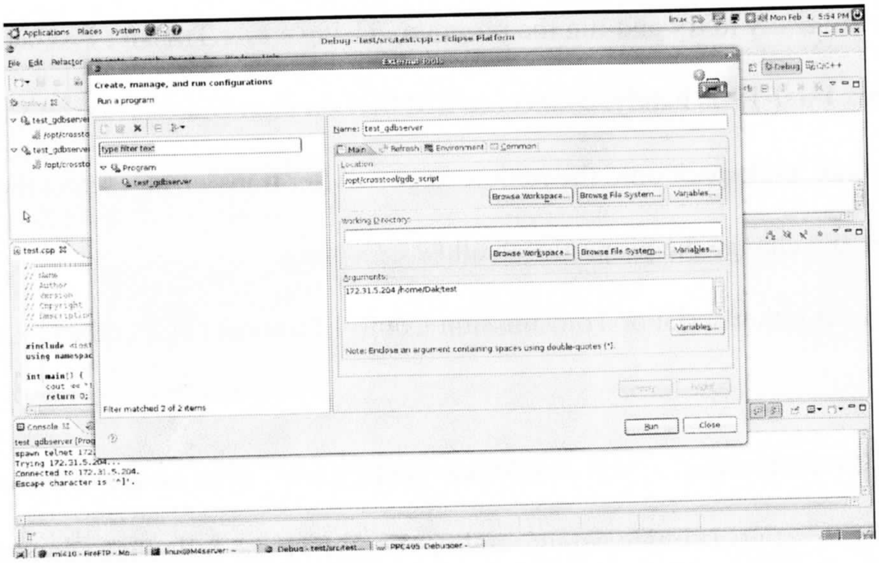
Now let's set up the GDB (GNU Debugger) Server, that will run on the remote target:

Create a new Program

Name: test_gdbserver

Location: /opt/crosstool/gdb_script

Arguments: <remote ip> <remote_location>/test



Name: test_gdbserver

Location: /opt/crosstool/gdb_script

Working Directory:

Arguments: 172.31.5.204 /home/Dak/test

Note: Enclose an argument containing spaces using double-quotes (").

```
spawn telnet 172.
Trying 172.31.5.204...
Connected to 172.31.5.204.
Escape character is '^]'.
```

Figure 2: GDB server setup in Eclipse

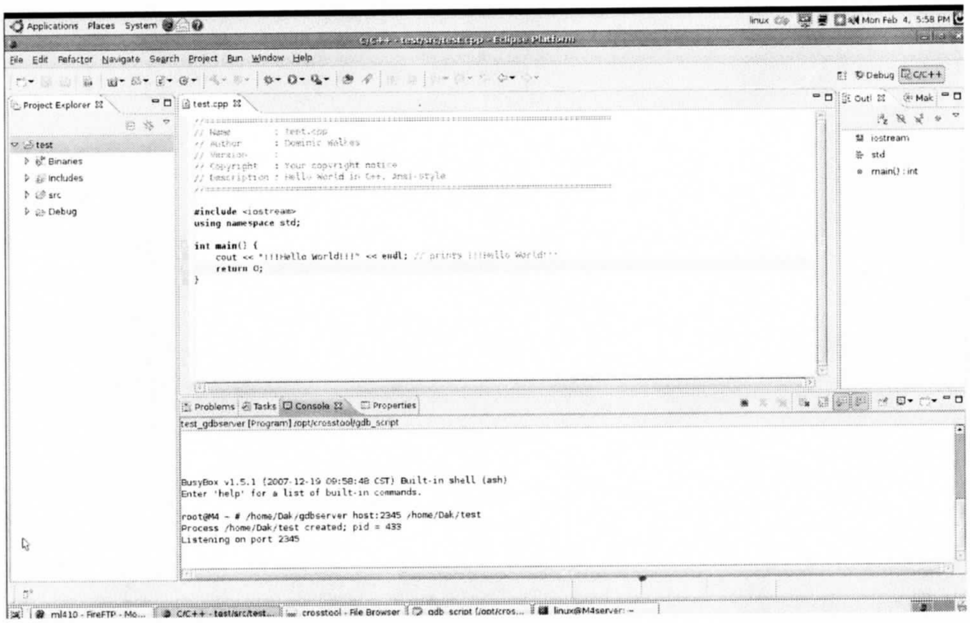
In this case, my remote target was 172.31.5.204 and the program was stored to "/home/Dak/test"

Note!! On tools rev 0.0.3 the following must be replaced in /opt/crosstool/gdb_script:

```
#!/usr/bin/expect -f
if {$argc==0} {
    send_user "usage: $argv0 IP_ADDRESS Prg_to_Debug\n"
    exit  }
set timeout -1
set ip [lindex $argv 0]
set login "root\r"
#set passwd "Dak\r"
set prg [lindex $argv 1]
set cmd "/home/Dak/gdbserver host:2345 $prg\r"
spawn telnet $ip
expect "M4 login: "
```

```
send $login
#expect "Password: "
#send $passwd
expect "root@M4 ~ # "
send $cmd
expect "Listening on port 2345"
expect "root@M4 ~ # "
send "exit\r"
```

Select Run - > you should see:


Applications Places System
linux Mon Feb 4, 5:58 PM
C/C++ - test/src/test.cpp - Eclipse Platform
File Edit Refactor Navigate Search Project Run Window Help
Debug C/C++
Project Explorer
test
Binaries
Includes
src
Debug
test.cpp
Outl Mak
iostream
std
main() : int
Problems Tasks Console Properties
ml410 - FireFTP - Mo... C/C++ - test/src/test... crosstool - File Browser gdb script (/opt/cros... linux@M4server: ~

```
//==========================================================================
// Name        : test.cpp
// Author      : Dominic Walkes
// Version     :
// Copyright   : Your copyright notice
// Description : Hello World in C++, Ansi-style
//==========================================================================

#include <iostream>
using namespace std;

int main() {
    cout << "!!!Hello World!!!" << endl; // prints !!!Hello World!!!
    return 0;
}
```

```
test_gdbserver [Program] /opt/crosstool/gdb_script

BusyBox v1.5.1 (2007-12-19 09:58:48 CST) Built-in shell (ash)
Enter 'help' for a list of built-in commands.

root@M4 ~ # /home/Dak/gdbserver host:2345 /home/Dak/test
Process /home/Dak/test created; pid = 433
Listening on port 2345
```

Figure 3: console checkup

In the console, make sure it says “Listening on port 2345” – if it does not debugging will not work! Now, back in the eclipse environment select Run->Open Debug Dialog

Create a new “C/C++ Local Application”

In the Main Tab-Name: Test Debug

Project: Test

C/C++ Application: Browse to <workspace_location>/test/Debug and select “test”

In the Debugger Tab-Debugger: gdbserver Debugger

MainTab-GDB Debugger: /opt/crosstool/PPC405_Debugger/bin/powerpc-405-linux-gnu-gdb

GDB command file: /opt/crosstool/PPC405_Debugger/gdb_setupShared Libraries:

Add -> /opt/crosstool/ml410_rfs

Connection:Type: TCP

Host Name: the remote location – 172.31.5.204 for example

Port Number 2345

Select RUN and you should be good to go!

CHAPTER 3

OBJECTIVE AND DELIVERABLE OF THE PROJECT

3.1 The objective of the project

M4 is the controller which will collect and store, in real time, diagnostic information for all internal components of the display for retrieval by the display control system. It is an embedded Linux based platform. The internal components which are stored in M4 Linux-kernel space diagnostic data is polled to the Linux-user Space diagnostic database using Application Program Interface(API).Project is to build the Linux-User Space Diagnostic database to generate the diagnostic notifications using the data provided from M4-Linux Kernel space.

Internal components polled for diagnostic data include:

- Controller status information (similar to the Get Status command of the control channel)
- Pixel failures
- Calibration data
- Component temperatures
- Display cooling assemblies along with on/off/speed control
- Power supply
- Display peripherals (i.e. temperature sensor, light detector, etc…)
- Other internal components required for display operation (i.e. MLC's, DD's, etc…)

The controller will provide diagnostic data for failed components only or a full status as defined by the requesting command. Diagnostic information will be returned in an Extensible Markup Language (XML) style of document to the display control application as part of the diagnostic data, a representation of the display component configuration shall be returned for use by the control application. (i. e. locations controlled by an MLC, area powered by a power supply, etc…).

The following is the plan to build the user level diagnostic database based on the index available from API and this Data is available at the Kernel level Diagnostic Database.

3.1.1 Black Box Diagnostic Database

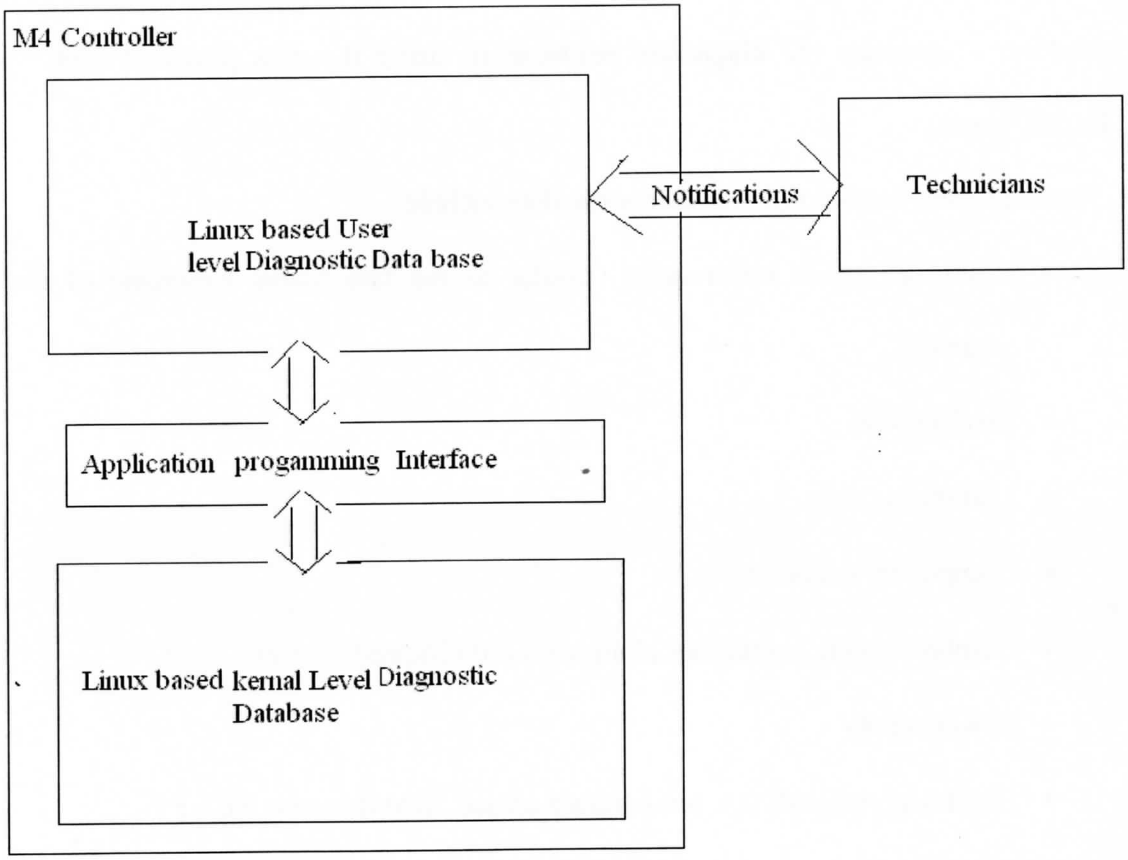

M4 Controller
Linux based User level Diagnostic Data base
Notifications
Technicians
Application progamming Interface
Linux based kernal Level Diagnostic Database

Figure 4: Black Box Diagnostic Database

3.1.2 Proposed M3/M4 Diagnostics

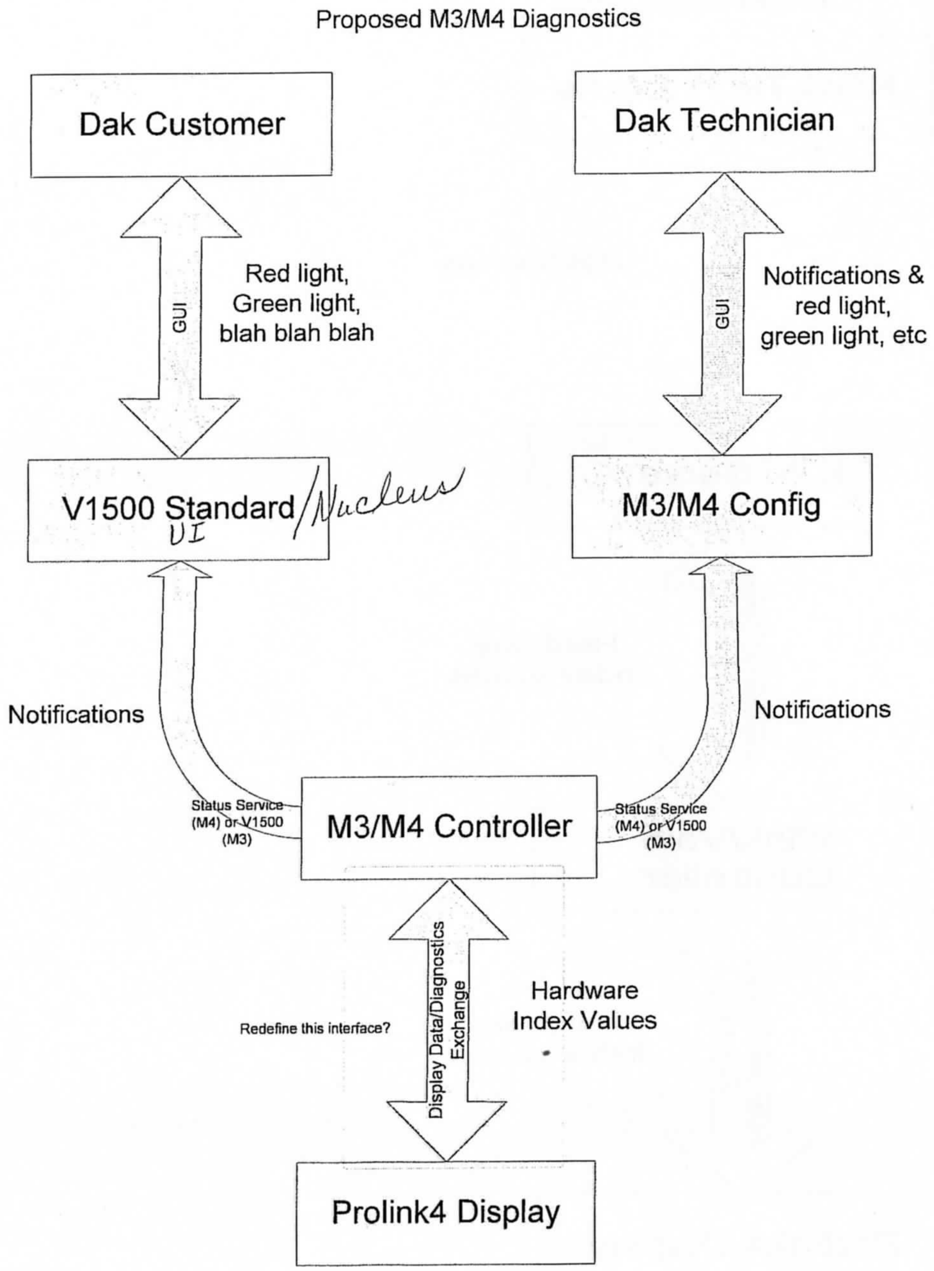


Figure 5: Proposed M3/M4 Diagnostics

3.1.3 Current Diagnostic Database

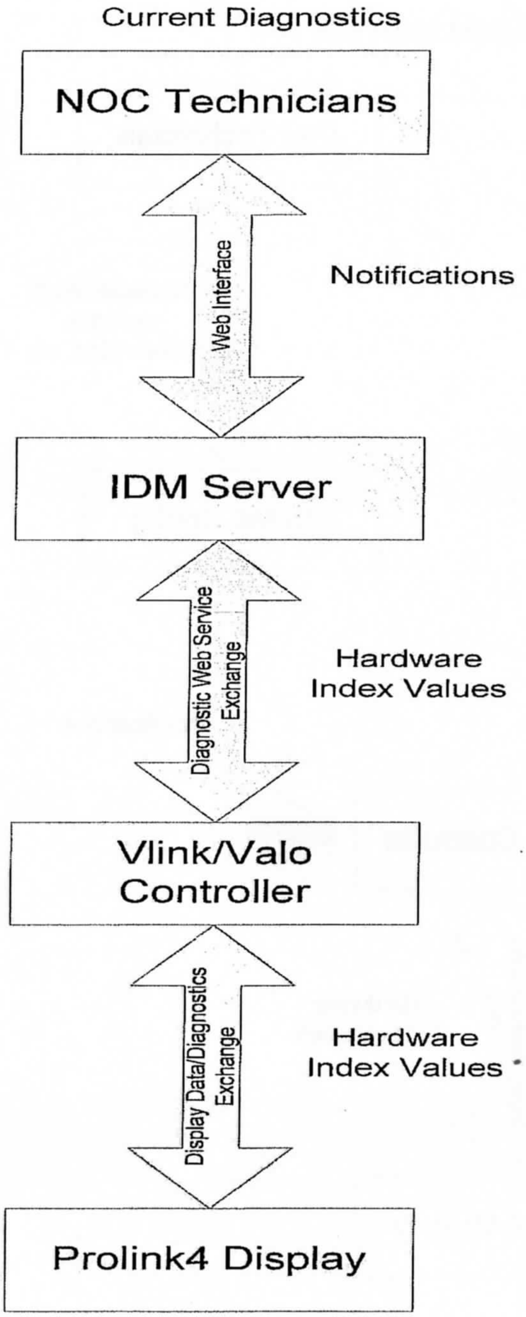


Figure 6: Current Diagnostic Database

3.2 Statement of the problem

Below are the current Diagnostic notifications generating functionality for controllers at server level. The above diagram is a designed diagnostic database based on this current structure.

3.2.1 Functionality of Controller

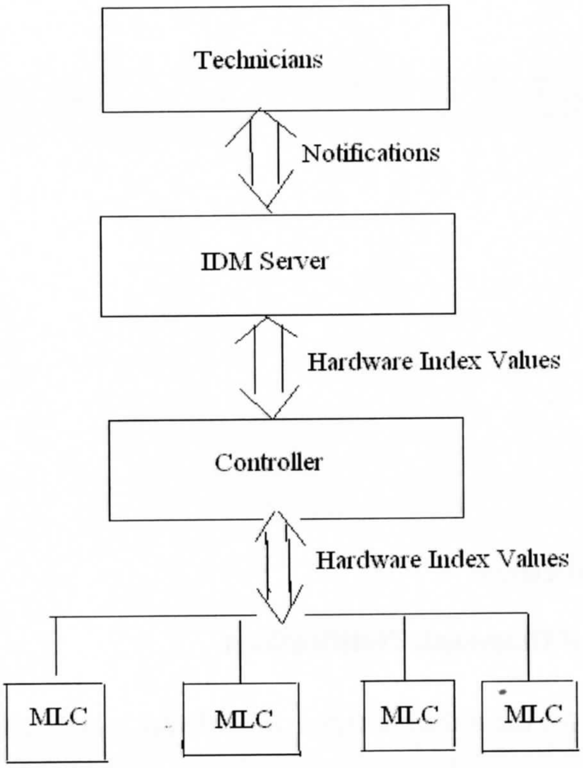
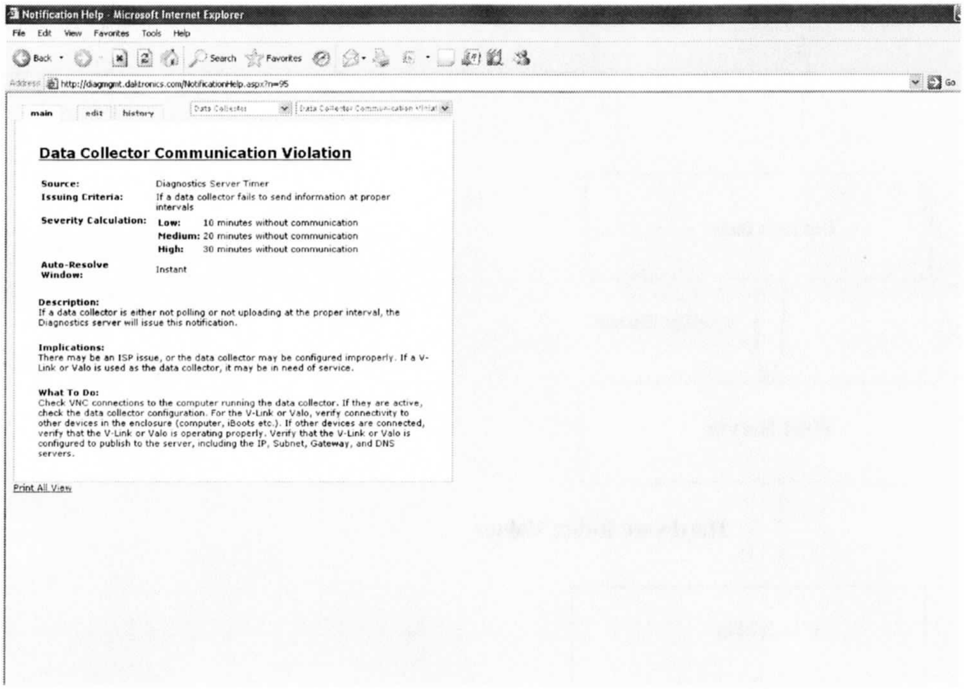


Figure 7: Functionality of Controller

The following screen shot is a sample notification generated at the server level diagnostic database site to show the issues in the controller. Similarly, we need to generate the notification for the M4 Controller in the future after building the database on the M4 controller. These notifications would be useful to store in the database to generate notifications. The difference is the Intelligent Device Management (IDM) is Server side Database and Database which we are trying to build is on M4 controller.

3.2.2. Sample Notification at Server level


Notification Help - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://diagmgmt.daktronics.com/NotificationHelp.aspx?n=95
Go
main edit history
Data Collector
Data Collector Communication Violat
Print All View

Data Collector Communication Violation

Source:	Diagnostics Server Timer
Issuing Criteria:	If a data collector fails to send information at proper intervals
Severity Calculation:	**Low:** 10 minutes without communication **Medium:** 20 minutes without communication **High:** 30 minutes without communication
Auto-Resolve Window:	Instant

Description:
If a data collector is either not polling or not uploading at the proper interval, the Diagnostics server will issue this notification.

Implications:
There may be an ISP issue, or the data collector may be configured improperly. If a V-Link or Valo is used as the data collector, it may be in need of service.

What To Do:
Check VNC connections to the computer running the data collector. If they are active, check the data collector configuration. For the V-Link or Valo, verify connectivity to other devices in the enclosure (computer, iBoots etc.). If other devices are connected, verify that the V-Link or Valo is operating properly. Verify that the V-Link or Valo is configured to publish to the server, including the IP, Subnet, Gateway, and DNS servers.

Figure 8: Screenshot of sample notification

3.2.3 The Matrix representation of Diagnostic Notifications

The following is the matrix for the different types of hardware and diagnostic notifications which are generated when the display problem and other problems occur in the M4 controller and are stored in the diagnostic database on the server side. Based on these notifications we need to build the notifications for the M4 Diagnostic Database. In order to do that we need to filter understand and store the M4 controller notification in the Controller

User Level Database.

V-Link 1500 DVI ProLink4	Control Enclosure Open
	Light Sensor Violation
	Non-DVI Signal Source
	Too Bright at Night!
	V-Link Not Receiving Video
	V-Link Over Temperature
	V-Link Thermal Dimming
V-Link 1500 DVI VMAX4	Control Enclosure Open
	Light Sensor Violation
	Non-DVI Signal Source
	Security Sensor 1 (Catwalk Back A) Tripped
	Security Sensor 2 (Catwalk Back B) Tripped
	Security Sensor 3 (Catwalk Back A) Tripped
	Security Sensor 4 (Catwalk Back B) Tripped
	Security Sensor 5 (Ladder Top) Tripped
	Security Sensor 6 (Ladder Bottom) Tripped
	Too Bright at Night!
	V-Link Not Receiving Video
	V-Link Over Temperature

ValoPlay ProLink4	Control Enclosure Open
	Error Downloading Alerts Schedule from Visiconn
	Error Downloading Content from Visiconn
	Error Downloading Schedule from Visiconn
	Error Uploading Error Log to Visiconn
	Error Uploading Runtime Log to Visiconn
	Light Sensor Violation
	Missed Scheduled Poll with Visiconn
	No Content Scheduled
	Playback Error
	Schedule Abnormality Detected
	SD Card Failure
	Security Failure with Visiconn
	Too Bright at Night!
	Valo Over Temperature
	Valo Running on Battery Backup
	Valo Thermal Dimming
	ValoPlay Reset
ValoPlay VMAX4	Control Enclosure Open
	Error Downloading Alerts Schedule from Visiconn

	Error Downloading Content from Visiconn
	Error Downloading Schedule from Visiconn
	Error Uploading Error Log to Visiconn
	Error Uploading Runtime Log to Visiconn
	Light Sensor Violation
	Missed Scheduled Poll with Visiconn
	No Content Scheduled
	Playback Error
	Schedule Abnormality Detected
	SD Card Failure
	Security Failure with Visiconn
	Too Bright at Night!
	Valo Over Temperature
	Valo Running on Battery Backup
	Valo Thermal Dimming
	ValoPlay Reset
V-Net 4	Data Error
	Error Downloading Alerts Schedule from Visiconn
	Error Downloading Content from Visiconn
	Error Downloading Schedule from Visiconn
	Error Uploading Error Log to Visiconn

	Error Uploading Runtime Log to Visiconn
	File Duration Exceeded
	Missed Scheduled Poll with Visiconn
	No Content Scheduled
	Playback Error
	Player Manager Not Running
	Player Manager Reset
	Player Not Running
	Unable to Activate Schedule
Data Collector	Data Collector Communication Violation
DD 4030	DD Communication Failure
	DD Diagnostic Data Violation
	DD Display Data Violation
	DD Input Signal Not Locked
	DD Reset
	DD Thermal Dimming
	Too Bright at Night!
DD 404X	DD Communication Failure
	DD Diagnostic Data Violation
	DD Display Data Violation
	DD PCB Over Temperature

	DD Input Signal Not Locked
	DD Reset
	DD Startup Error - FPGA Not Configured
	DD Startup Error - Memory Self Test Failed
	DD Startup Error - Test Pattern Blanking Failed
	DD Startup Error - Translation Table Checksum Failed
	DD Startup Error - Translation Table Not Enabled
	DD Thermal Dimming
MLC 3050	MLC Diagnostic Data Violation
	MLC Display Clock Violation
	MLC Display Data Violation
	MLC Reset
MLC 4050	MLC Diagnostic Data Violation
	MLC Primary Signal Display Data Violation
	MLC Primary Signal Out of Sync
	MLC Reset
	MLC Secondary Signal Display Data Violation
	MLC Secondary Signal Out of Sync

MOD 2X14	Module Diagnostic Data Violation
	Module Display Data Violation
	Module Over Temperature
MOD HD-16	Module Diagnostic Data Serial Violation
	Module Diagnostic Data Violation
	Module Display Data Violation
	Module Over Temperature
Sony FWD-50PX2	Screen Communication Failure
	Screen in Stand By Mode
	Screen Over Temperature
Sony FWD-50PX3	Screen Communication Failure
	Screen in Stand By Mode
	Screen Over Temperature

3.3 The Deliverable of the project

3.3.1Overview of Data

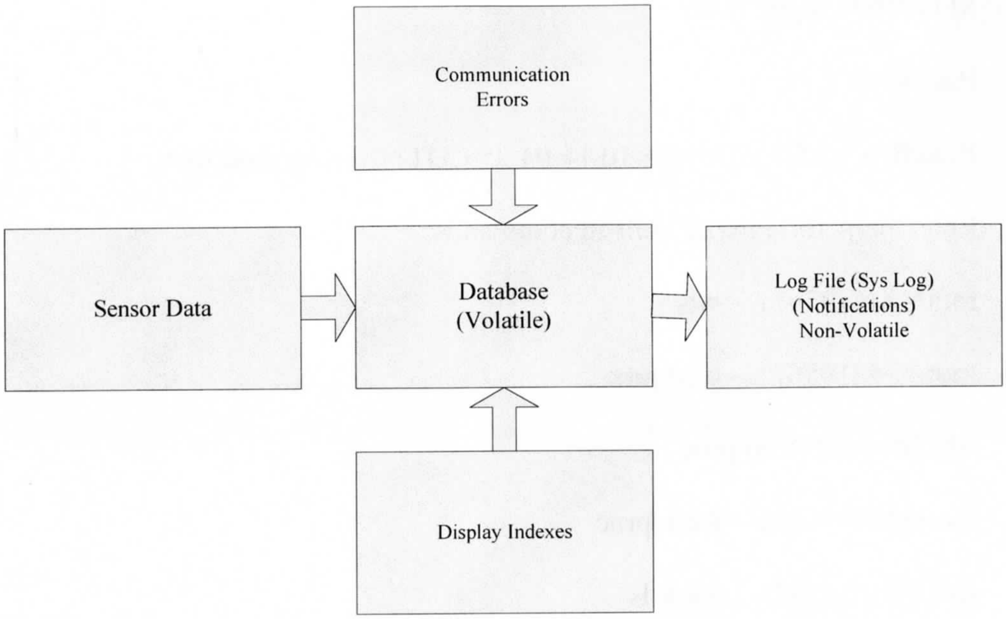


Figure 9: overview of Data

3.3.2Telenet of M4 Controller Display Data

The following is the M4 Controller display data retrieved through the telnet the process file system at the terminal in the Ubuntu. To get basic idea of the how the data is stored in the M4 Controller.

```
nkota@D6678-linux:~$ telnet m4195671

Trying 172.31.5.69...

Connected to m4195671.daktronics.lan.

Escape character is '^]'.

========================
M4 PPC Linux 2.6.21

Linux/ppc 2.6.21

======================

M4195671 login: root

Password:

BusyBox v1.5.1 (2008-06-10 14:04:35 CDT) Built-in shell (ash)

Enter 'help' for a list of built-in commands.

root@M4195671 ~ # ls

root@M4195671 ~ # cd proc

-sh: cd: can't cd to proc

root@M4195671 ~ # cd /proc

root@M4195671 /proc # ls

1      126    21346  242    285    4      bus        fs          loadavg
partitions  tty

10     134    21347  261    286    5      cmdline    interrupts
locks    scsi    uptime

100    137    21822  266    287    6      cpuinfo    iomem
meminfo    self    version

103    141    21830  278    288    7      devices    ioports     misc
slabinfo    vmstat

107    14823  223    279    289    77     diskstats  irq
modules    stat    zoneinfo
```

```
        11      152     226     280     3       8       driver    kallsyms    mounts
sys
        111     2       227     281     34      9       execdomains  kcore    mtd
sysvipc
        12      202     228     282     35      buddyinfo    filesystems  kmsg
net     timer_list
        root@M4195671 /proc # cd /driver
        -sh: cd: can't cd to /driver
        root@M4195671 /proc # cd driver
        root@M4195671 /proc/driver # ls
        M4Hdwr       display_driver  light      mpegdecode    prolink4     temp
        compositor   frame_control   lm86       pl4_diag      rtc
        root@M4195671 /proc/driver # ls
        M4Hdwr       display_driver  light      mpegdecode    prolink4     temp
        compositor   frame_control   lm86       pl4_diag      rtc
        root@M4195671 /proc/driver # cd pl4_diag
        root@M4195671 /proc/driver/pl4_diag # ls
        debug  port_0  port_1  status
        root@M4195671 /proc/driver/pl4_diag # cd port_0
        root@M4195671 /proc/driver/pl4_diag/port_0 # ls
        isac_0 mlc_0  mod_1  mod_11 mod_13 mod_15 mod_17 mod_3  mod_5  mod_7
mod_9
```

```
isac_1  mod_0  mod_10  mod_12  mod_14  mod_16  mod_2  mod_4  mod_6  mod_8

root@M4195671 /proc/driver/pl4_diag/port_0 # cd mlc_0

root@M4195671 /proc/driver/pl4_diag/port_0/mlc_0 # ls

created  dynamic  generic  static   various

root@M4195671 /proc/driver/pl4_diag/port_0/mlc_0 # cd created

-sh: cd: can't cd to created

root@M4195671 /proc/driver/pl4_diag/port_0/mlc_0 # cat created

PDA2

0000root@M4195671 /proc/driver/pl4_diag/port_0/mlc_0 # cat PDA@

cat: PDA@: No such file or directory

root@M4195671 /proc/driver/pl4_diag/port_0/mlc_0 # cat PDA2

cat: PDA2: No such file or directory

root@M4195671 /proc/driver/pl4_diag/port_0/mlc_0 # cd dynamic

-sh: cd: can't cd to dynamic

root@M4195671 /proc/driver/pl4_diag/port_0/mlc_0 # cat dynamic

PDA2

0030

0840

0000

0021

0031

ffff
```

```
ffff
0000
0001
00ff
0000
010a
0001
ffff
ffff
ffff
05dc
2db4
ffff
0001
0000
003c
0000
0001
0002
e26d
0000
0001
0000
```

```
724a
0000
0001
0002
e26d
0000
0001
00ff
ffff
ffff
0006
0004
0059
0009
5edd
0003
ed15
0002
da62
0002
root@M4195671 /proc/driver/pl4_diag/port_0/mlc_0 # ls
created  dynamic  generic  static   various
root@M4195671 /proc/driver/pl4_diag/port_0/mlc_0 # cat generic
```

```
gd_proc generic

port 0 type 2 num 0

root@M4195671 /proc/driver/pl4_diag/port_0/mlc_0 # cat port 0

cat: port: No such file or directory

cat: 0: No such file or directory

root@M4195671 /proc/driver/pl4_diag/port_0/mlc_0 # cat static

PDA2

0031

1302

0020

0033

0054

0039

0006

0001

0000

7911

c7bd

002f

0001

005f

0003

0015
```

```
0015
0015
0015
0015
0015
ffff
ffff
0c5f
0c5f
0c5f
0c5f
0c5f
0c5f
0c5f
0c5f
0c5f
0c5f
0c5f
0c5f
ffff
ffff
ffff
ffff
```

```
0000

0000

29ad

0046

0016

0000

0000

0000

0000

0000

0007
```

3.3.3 Flow of Data

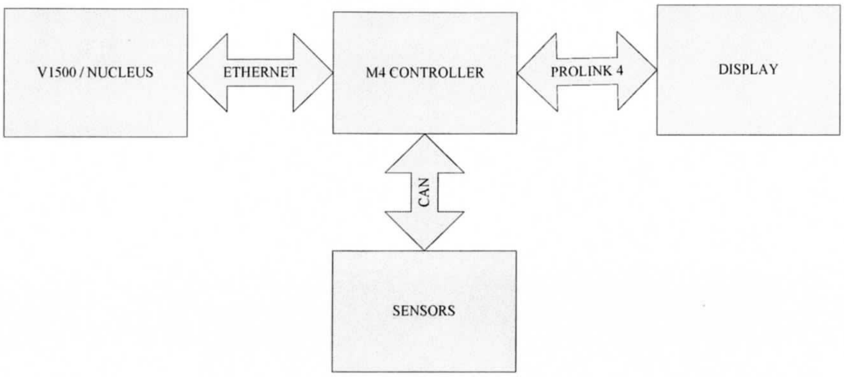


Figure 10: Figure-10 Flow of Data

3.3.4 Telnet of M4 Controller Sensor Data

The following is the M4 controller Sensor data retrieved through the telnet the process file system at the terminal in the Ubuntu to get basic idea of the how the data is stored in the M4 Controller.

```
nkota@D6678-linux:~$ telnet mm4
 Trying 10.6.70.131...
 Connected to mm4.daktronics.lan.
 Escape character is '^]'.
 ======================
 M4 PPC Linux 2.6.21
```

```
Linux/ppc 2.6.21

=======================

MM4 login: root

Password:


BusyBox v1.5.1 (2008-06-10 14:04:35 CDT) Built-in shell (ash)

Enter 'help' for a list of built-in commands.


root@MM4 ~ # ls

discoveryd

root@MM4 ~ # cd /proc

root@MM4 /proc # ls

1        16866     21582     4         fs          net

10       16951     216       5         interrupts  partitions

101      16955     217       6         iomem       scsi

11       16956     259       7         ioports     self

12       16957     2687      8         irq         slabinfo

126      16958     2688      9         kallsyms    stat

129      16963     2967      buddyinfo kcore       sys

152      16968     2975      bus       kmsg        sysvipc

156      16969     3         cmdline   loadavg     timer_list

164      16970     34        cpuinfo   locks       tty

16673    183       35        devices   meminfo     uptime
```

```
16792      2        354       diskstats   misc       version
16793      211      363       driver      modules    vmstat
16794      214      37        execdomains mounts     zoneinfo
168        21578    38        filesystems mtd
root@MM4 /proc # cd light
-sh: cd: can't cd to light
root@MM4 /proc # cd /light
-sh: cd: can't cd to /light
root@MM4 /proc # cd /driver
-sh: cd: can't cd to /driver
root@MM4 /proc # cd driver
root@MM4 /proc/driver # ls
M4Hdwr         frame_control  mpegdecode    rtc
compositor     light          pl4_diag      temp
display_driver lm86           prolink4
root@MM4 /proc/driver # cd light
root@MM4 /proc/driver/light # ls
address  one     rev     ten     timer
root@MM4 /proc/driver/light # ls
address  one     rev     ten     timer
root@MM4 /proc/driver/light # cat address
0x42012root@MM4 /proc/driver/light # cd address
-sh: cd: can't cd to address
```

```
root@MM4 /proc/driver/light # cat one
one_minute_avg = 49
root@MM4 /proc/driver/light # cat rev
rev = 8
root@MM4 /proc/driver/light # cat ten
ten_minute_avg = 49
root@MM4 /proc/driver/light # cat timer
30
root@MM4 /proc/driver/light # cd ..
root@MM4 /proc/driver # ls
M4Hdwr       display_driver  light        mpegdecode    prolink4      temp
compositor    frame_control   lm86         pl4_diag      rtc
root@MM4 /proc/driver # cd pl4_diag
root@MM4 /proc/driver/pl4_diag # ls
debug   port_0  port_1  status
root@MM4 /proc/driver/pl4_diag # cd port_0
root@MM4 /proc/driver/pl4_diag/port_0 # ls
mlc_0   mod_124  mod_152  mod_180  mod_208  mod_236  mod_264  mod_292
mod_32  mod_348  mod_376  mod_403  mod_431  mod_46   mod_488  mod_55   mod_83
mlc_1   mod_125  mod_153  mod_181  mod_209  mod_237  mod_265  mod_293
mod_320  mod_349  mod_377  mod_404  mod_432  mod_460  mod_489  mod_56   mod_84
mod_0   mod_126  mod_154  mod_182  mod_21   mod_238  mod_266  mod_294
mod_321  mod_35   mod_378  mod_405  mod_433  mod_461  mod_49   mod_57   mod_85
```

```
          mod_1   mod_127 mod_155 mod_183 mod_210 mod_239 mod_267 mod_295
mod_322 mod_350 mod_379 mod_406 mod_434 mod_462 mod_490 mod_58  mod_86
          mod_10  mod_128 mod_156 mod_184 mod_211 mod_24  mod_268 mod_296
mod_323 mod_351 mod_38  mod_407 mod_435 mod_463 mod_491 mod_59  mod_87
          mod_100 mod_129 mod_157 mod_185 mod_212 mod_240 mod_269 mod_297
mod_324 mod_352 mod_380 mod_408 mod_436 mod_464 mod_492 mod_6   mod_88
          mod_101 mod_13  mod_158 mod_186 mod_213 mod_241 mod_27  mod_298
mod_325 mod_353 mod_381 mod_409 mod_437 mod_465 mod_493 mod_60  mod_89
          mod_102 mod_130 mod_159 mod_187 mod_214 mod_242 mod_270 mod_299
mod_326 mod_354 mod_382 mod_41  mod_438 mod_466 mod_494 mod_61  mod_9
          mod_103 mod_131 mod_16  mod_188 mod_215 mod_243 mod_271 mod_3
mod_327 mod_355 mod_383 mod_410 mod_439 mod_467 mod_495 mod_62  mod_90
          mod_104 mod_132 mod_160 mod_189 mod_216 mod_244 mod_272 mod_30
mod_328 mod_356 mod_384 mod_411 mod_44  mod_468 mod_496 mod_63  mod_91
          mod_105 mod_133 mod_161 mod_19  mod_217 mod_245 mod_273 mod_300
mod_329 mod_357 mod_385 mod_412 mod_440 mod_469 mod_497 mod_64  mod_92
          mod_106 mod_134 mod_162 mod_190 mod_218 mod_246 mod_274 mod_301
mod_33  mod_358 mod_386 mod_413 mod_441 mod_47  mod_498 mod_65  mod_93
          mod_107 mod_135 mod_163 mod_191 mod_219 mod_247 mod_275 mod_302
mod_330 mod_359 mod_387 mod_414 mod_442 mod_470 mod_499 mod_66  mod_94
          mod_108 mod_136 mod_164 mod_192 mod_22  mod_248 mod_276 mod_303
mod_331 mod_36  mod_388 mod_415 mod_443 mod_471 mod_5   mod_67  mod_95
```

mod_109 mod_137 mod_165 mod_193 mod_220 mod_249 mod_277 mod_304 mod_332 mod_360 mod_389 mod_416 mod_444 mod_472 mod_50 mod_68 mod_96

mod_11 mod_138 mod_166 mod_194 mod_221 mod_25 mod_278 mod_305 mod_333 mod_361 mod_39 mod_417 mod_445 mod_473 mod_500 mod_69 mod_97

mod_110 mod_139 mod_167 mod_195 mod_222 mod_250 mod_279 mod_306 mod_334 mod_362 mod_390 mod_418 mod_446 mod_474 mod_501 mod_7 mod_98

mod_111 mod_14 mod_168 mod_196 mod_223 mod_251 mod_28 mod_307 mod_335 mod_363 mod_391 mod_419 mod_447 mod_475 mod_502 mod_70 mod_99

mod_112 mod_140 mod_169 mod_197 mod_224 mod_252 mod_280 mod_308 mod_336 mod_364 mod_392 mod_42 mod_448 mod_476 mod_503 mod_71

mod_113 mod_141 mod_17 mod_198 mod_225 mod_253 mod_281 mod_309 mod_337 mod_365 mod_393 mod_420 mod_449 mod_477 mod_504 mod_72

mod_114 mod_142 mod_170 mod_199 mod_226 mod_254 mod_282 mod_31 mod_338 mod_366 mod_394 mod_421 mod_45 mod_478 mod_505 mod_73

mod_115 mod_143 mod_171 mod_2 mod_227 mod_255 mod_283 mod_310 mod_339 mod_367 mod_395 mod_422 mod_450 mod_479 mod_506 mod_74

mod_116 mod_144 mod_172 mod_20 mod_228 mod_256 mod_284 mod_311 mod_34 mod_368 mod_396 mod_423 mod_451 mod_48 mod_507 mod_75

mod_117 mod_145 mod_173 mod_200 mod_229 mod_257 mod_285 mod_312 mod_340 mod_369 mod_397 mod_424 mod_452 mod_480 mod_508 mod_76

mod_118 mod_146 mod_174 mod_201 mod_23 mod_258 mod_286 mod_313 mod_341 mod_37 mod_398 mod_425 mod_453 mod_481 mod_509 mod_77

```
    mod_119 mod_147 mod_175 mod_202 mod_230 mod_259 mod_287 mod_314
mod_342 mod_370 mod_399 mod_426 mod_454 mod_482 mod_51 mod_78
    mod_12  mod_148 mod_176 mod_203 mod_231 mod_26  mod_288 mod_315
mod_343 mod_371 mod_4   mod_427 mod_455 mod_483 mod_510 mod_79
    mod_120 mod_149 mod_177 mod_204 mod_232 mod_260 mod_289 mod_316
mod_344 mod_372 mod_40  mod_428 mod_456 mod_484 mod_511 mod_8
    mod_121 mod_15  mod_178 mod_205 mod_233 mod_261 mod_29  mod_317
mod_345 mod_373 mod_400 mod_429 mod_457 mod_485 mod_52  mod_80
    mod_122 mod_150 mod_179 mod_206 mod_234 mod_262 mod_290 mod_318
mod_346 mod_374 mod_401 mod_43  mod_458 mod_486 mod_53  mod_81
    mod_123 mod_151 mod_18  mod_207 mod_235 mod_263 mod_291 mod_319
mod_347 mod_375 mod_402 mod_430 mod_459 mod_487 mod_54  mod_82
    root@MM4 /proc/driver/pl4_diag/port_0 # cd mlc_0
    root@MM4 /proc/driver/pl4_diag/port_0/mlc_0 # ls
    created  dynamic  generic  static  various
    root@MM4 /proc/driver/pl4_diag/port_0/mlc_0 # cat dynamic
    PDA2
    0000root@MM4 /proc/driver/pl4_diag/port_0/mlc_0 # cat created
    PDA2
    0000root@MM4 /proc/driver/pl4_diag/port_0/mlc_0 # cat generic
    gd_proc generic
    port 0 type 2 num 0
    root@MM4 /proc/driver/pl4_diag/port_0/mlc_0 #
```

```
        _149 mod_177 mod_204 mod_232 mod_260 mod_289 mod_316 mod_344
mod_372 mod_40  mod_428 mod_456 mod_484 mod_511 mod_8
        mod_121 mod_15  mod_178 mod_205 mod_233 mod_261 mod_29  mod_317
mod_345 mod_373 mod_400 mod_429 mod_457 mod_485 mod_52  mod_80
        mod_122 mod_150 mod_179 mod_206 mod_234 mod_262 mod_290 mod_318
mod_346 mod_374 mod_401 mod_43  mod_458 mod_486 mod_53  mod_81
        mod_123 mod_151 mod_18  mod_207 mod_235 mod_263 mod_291 mod_319
mod_347 mod_375 mod_402 mod_430 mod_459 mod_487 mod_54  mod_82
        root@MM4 /proc/driver/pl4_diag/port_0 # cd mlc_0
        root@MM4 /proc/driver/pl4_diag/port_0/mlc_0 # ls
        created  dynamic  generic  static   various
        root@MM4 /proc/driver/pl4_diag/port_0/mlc_0 # cat dynamic
        PDA2
        0000root@MM4 /proc/driver/pl4_diag/port_0/mlc_0 # cat created
        PDA2
        0000root@MM4 /proc/driver/pl4_diag/port_0/mlc_0 # cat generic
        gd_proc generic
        port 0 type 2 num 0
        root@MM4 /proc/driver/pl4_diag/port_0/mlc_0 #
```

3.3.5 Flow Chart for Diagnostic Database

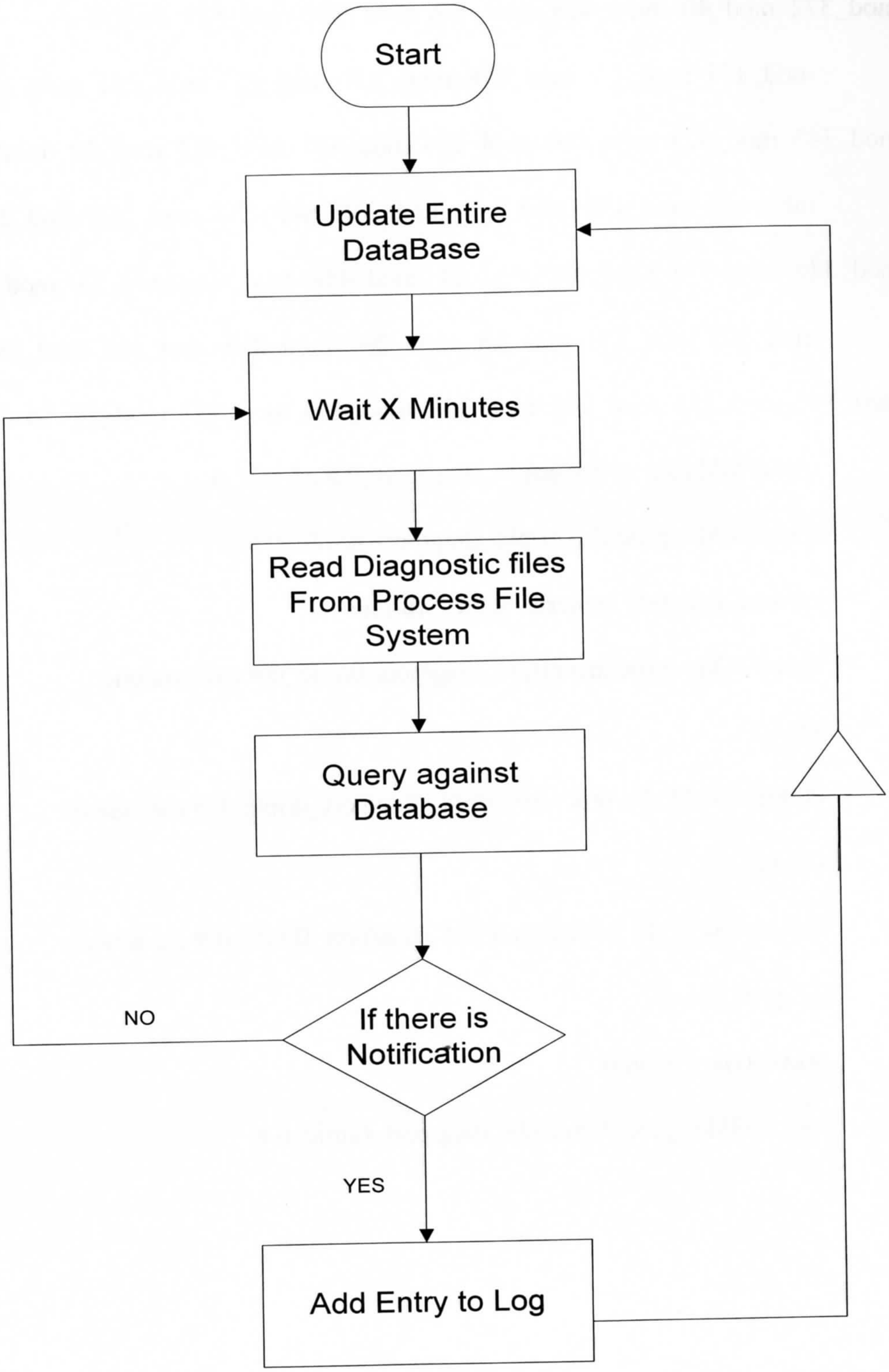


Figure 11: Flow Chart for Diagnostic Database

CHAPTER 4

SYSTEM DESIGN & RESULTS

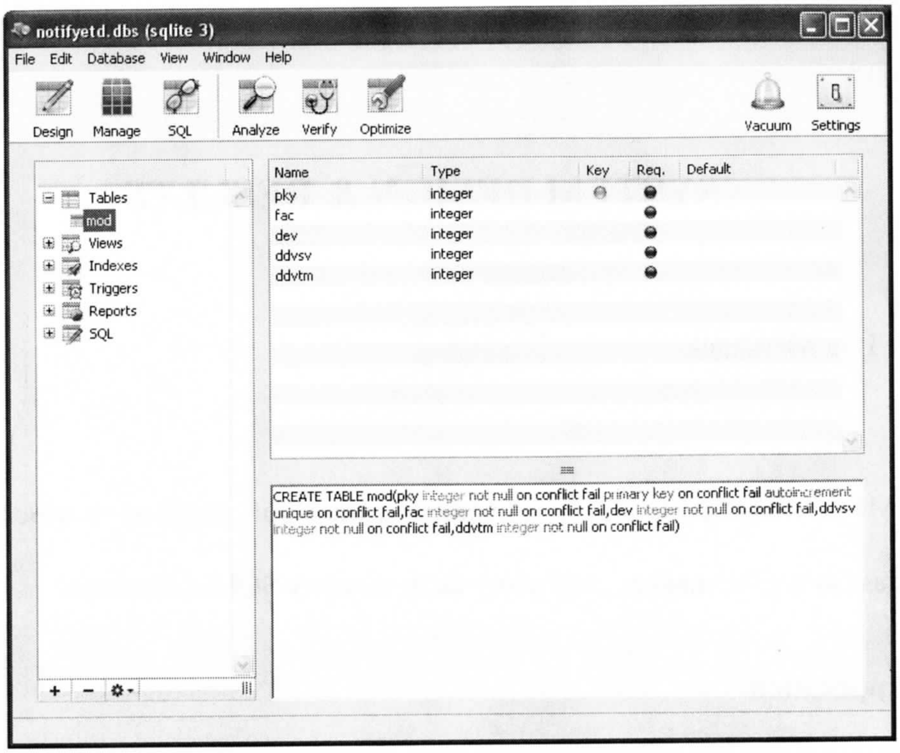
4.1 M4 Notify Log Messages

Abstract

The M4 notification log system is handled by the Notify application which uses the CNotifyLog class to create, manage, and query an in memory SQLite database.

4.1.1 Notify Application

Notify, the application used to manage this system, creates an SQLite database as shown by Figure 13. This database can be created on a disk using the "-f" switch. I will create an SQLite database named "notifyetd.dbs", i.e. notify extended database. If the "-f" switch is not specified the database name is ":memory:" which SQLite interprets as being in memory, i.e. it lasts only as long as the Notify application is running.



Name	Type	Key	Req.	Default
pky	integer	●	●	
fac	integer		●	
dev	integer		●	
ddvsv	integer		●	
ddvtm	integer		●	

```
CREATE TABLE mod(pky integer not null on conflict fail primary key on conflict fail autoincrement
unique on conflict fail,fac integer not null on conflict fail,dev integer not null on conflict fail,ddvsv
integer not null on conflict fail,ddvtm integer not null on conflict fail)
```

Figure 12: Screen Shot of sample SQLite Database in windows

4.1.2 NotifyLog Class

The CNotifyLog class builds the SQL table as shown in Figure 2. This class creates the table and populates it with the fixed data consisting of the following fields:

- pky - Primary Key (automatically generated)
- fac – Facility Code (defined in Table 1)
- dev – MOD Device (defined as Port times 1000 plus MOD index)

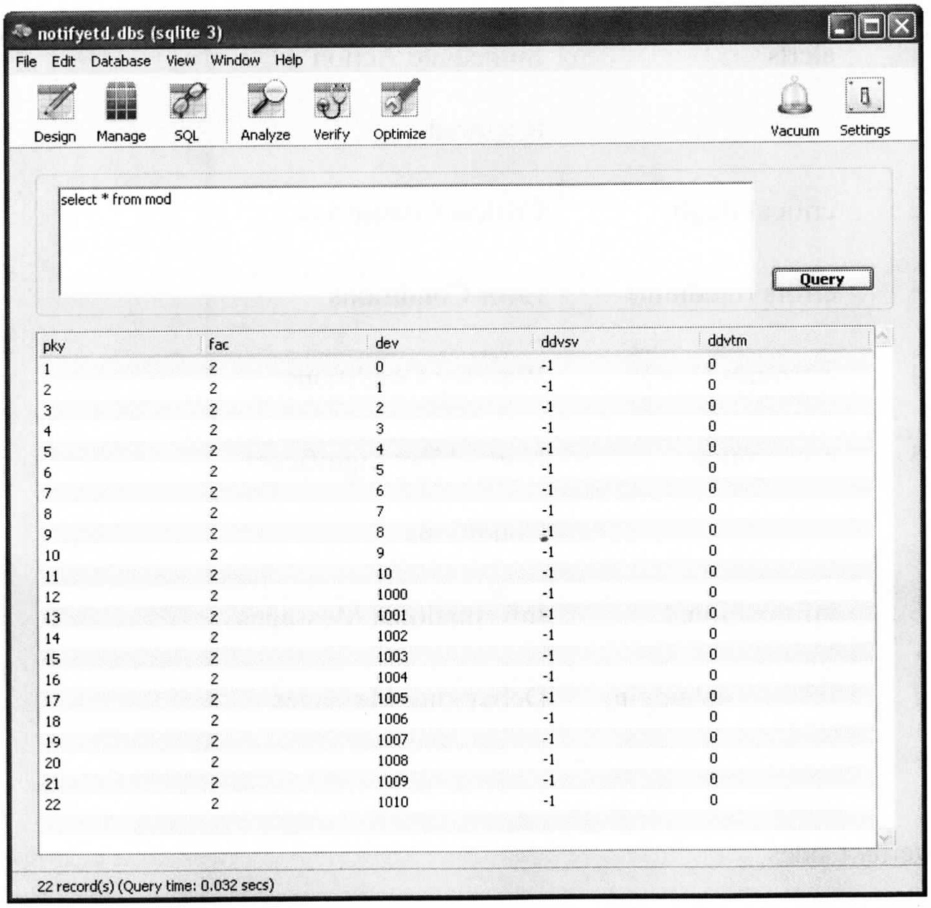
Variable data fields include the following:

- ddvsv – Diagnostics Data Violation Severity (severity is defined in Table 2 and follows the syslog convention)
- ddvtm – Unix Based Time Stamp When ddvsv Detected

Fixed field data is generated by the class logic which attempts to do an fopen on a file given by the following path:

- /proc/driver/pl4_diag/port_i/mod_j/static (for embedded Linux where i is the port number from 0 to 4 and j is the mod index from 0 to 1000)
- C:\proc\driver\pl4_diag\port_i\mod_j\static (for embedded Linux where i is the port number from 0 to 4 and j is the mod index from 0 to 1000)
- /home/nkota//proc/driver/pl4_diag/port_i/mod_j/static (for Ubuntu Linux where i is the port number from 0 to 4 and j is the mod index from 0 to 1000)

4.1.2 NotifyLog Class



select * from mod

pky	fac	dev	ddvsv	ddvtm
1	2	0	-1	0
2	2	1	-1	0
3	2	2	-1	0
4	2	3	-1	0
5	2	4	-1	0
6	2	5	-1	0
7	2	6	-1	0
8	2	7	-1	0
9	2	8	-1	0
10	2	9	-1	0
11	2	10	-1	0
12	2	1000	-1	0
13	2	1001	-1	0
14	2	1002	-1	0
15	2	1003	-1	0
16	2	1004	-1	0
17	2	1005	-1	0
18	2	1006	-1	0
19	2	1007	-1	0
20	2	1008	-1	0
21	2	1009	-1	0
22	2	1010	-1	0

22 record(s) (Query time: 0.032 secs)

Figure 13: Screen Shot of MOD Table

Facility	System	Description
0	SYS	System
1	CAN	CAN Sensors
2	DSP	Display
3	ETH	Ethernet

Table 1: Facility Codes

Severity Level	Keyword	Description
0	emergencies	System Unusable
1	alerts	Immediate Action Required
2	critical (high)	Critical Conditions
3	errors (medium)	Error Conditions
4	warnings (low)	Warning Conditions
5	notifications	Normal But Significant Conditions
6	informational	Informational Messages
7	debugging	Debugging Messages

Table 2: Severity Codes

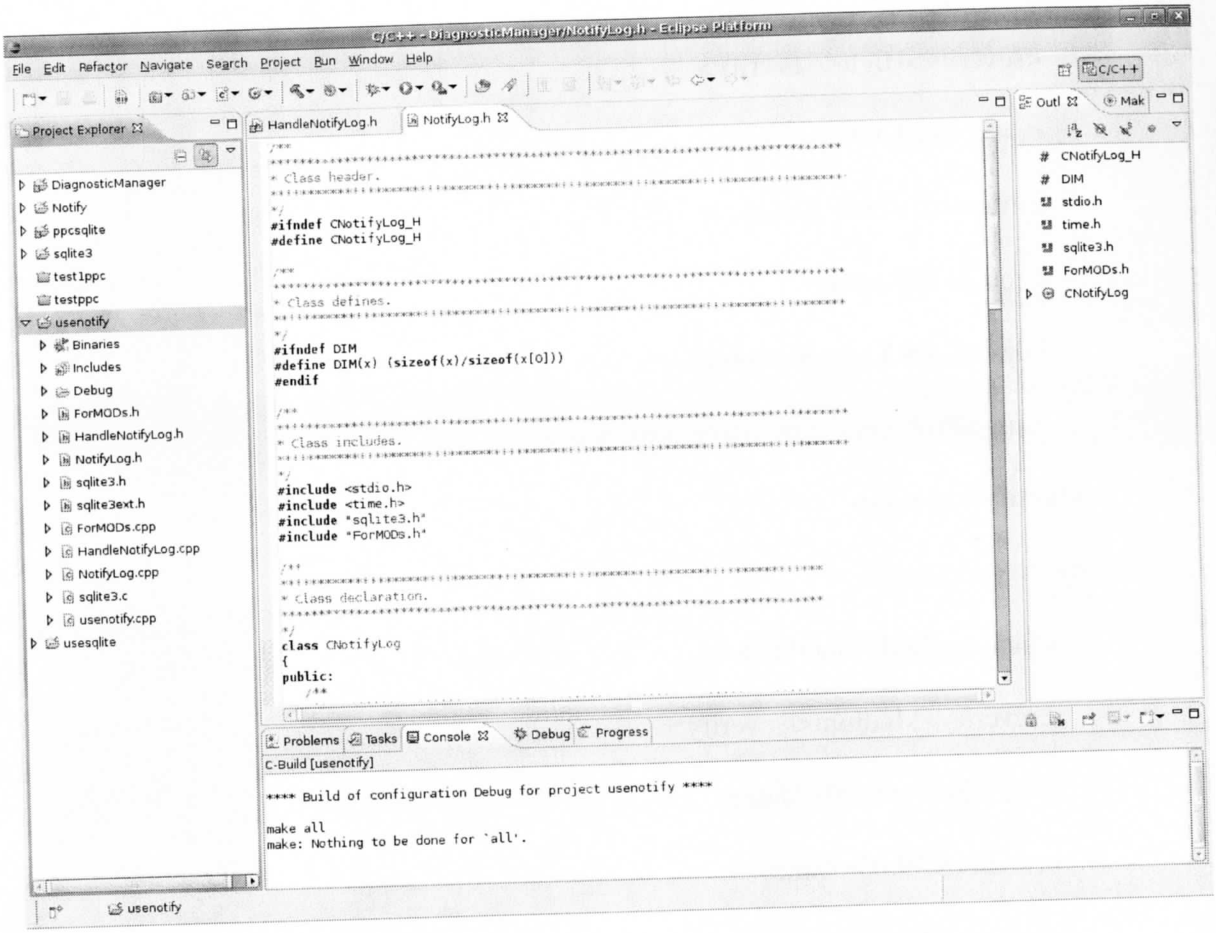
4.1.3 Notification XML

M4 Notify Log Messages

```
<notify-log-msg>
   <facility>CAN</facility>
   <severity>High</severity>
   <date>2008-09-03</date>
   <time>15:59:05</time>
   <zone>-300</zone>
   <dev>CAN Light Sensor</dev>
   <msg>Problem Communication</msg>
</notify-log-msg>
<notify-log-msg>
   <facility>DSP</facility>
   <severity>Medium</severity>
   <date>2008-09-03</date>
   <time>15:59:05</time>
   <zone>-300</zone>
   <dev>MOD 0</dev>
   <msg>Problem Module Diagnostics Data Violation</msg>
</notify-log-msg>
```

4.2 Screen Shots of Project implementation

4.2.1 The Diagnostic Manager Project built Screen Shot



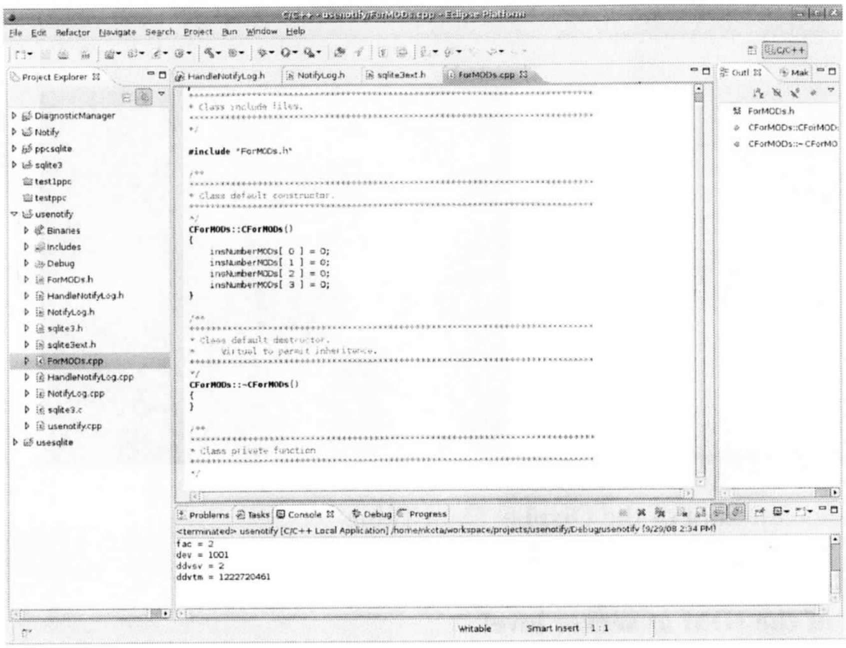
```
/**
*************************************************************************
* Class header.
*************************************************************************
*/
#ifndef CNotifyLog_H
#define CNotifyLog_H

/**
*************************************************************************
* Class defines.
*************************************************************************
*/
#ifndef DIM
#define DIM(x) (sizeof(x)/sizeof(x[0]))
#endif

/**
*************************************************************************
* Class includes.
*************************************************************************
*/
#include <stdio.h>
#include <time.h>
#include "sqlite3.h"
#include "ForMODs.h"

/**
*************************************************************************
* Class declaration.
*************************************************************************
*/
class CNotifyLog
{
public:
    /**
```

C-Build [usenotify]

```
**** Build of configuration Debug for project usenotify ****

make all
make: Nothing to be done for `all'.
```

Figure 14: Screen Shot of Diagnostic Manager Project built Screen Shot

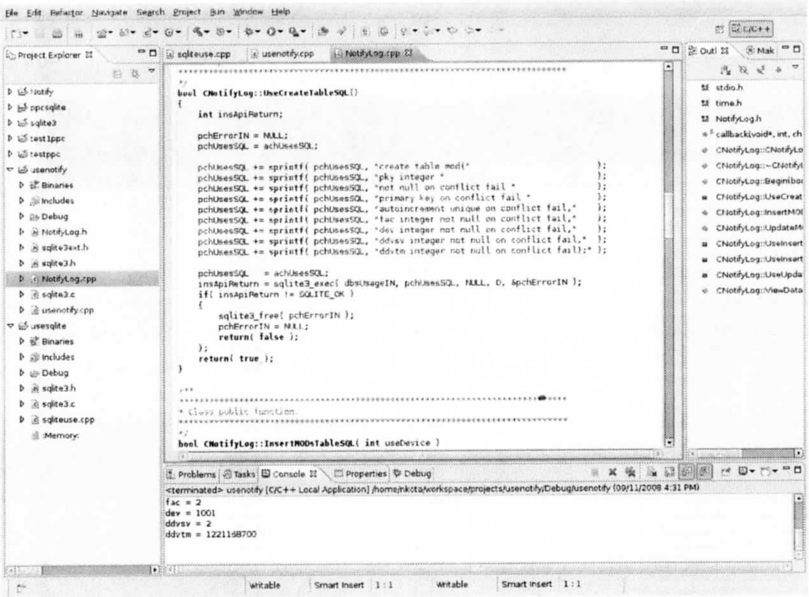
4.2.2 The Output of Diagnostic Manager Project



```
*************************************************************************
* Class include files.
*************************************************************************
*/

#include "ForMODs.h"

/**
*************************************************************************
* Class default constructor.
*************************************************************************
*/
CForMODs::CForMODs()
{
    insNumberMODs[ 0 ] = 0;
    insNumberMODs[ 1 ] = 0;
    insNumberMODs[ 2 ] = 0;
    insNumberMODs[ 3 ] = 0;
}

/**
*************************************************************************
* Class default destructor.
*     Virtual to permit inheritance.
*************************************************************************
*/
CForMODs::~CForMODs()
{
}

/**
*************************************************************************
* Class private function
*************************************************************************
*/
```

```
<terminated> usenotify [C/C++ Local Application] /home/nkota/workspace/projects/usenotify/Debug/usenotify (9/29/08 2:34 PM)
fac = 2
dev = 1001
ddvsv = 2
ddvtm = 1222720461
```

Figure 15: Screen shot of output of Diagnostic Manager Project



```
*************************************************************************
*/
bool CNotifyLog::UseCreateTableSQL()
{
    int insApiReturn;

    pchErrorIN = NULL;
    pchUsesSQL = achUsesSQL;

    pchUsesSQL += sprintf( pchUsesSQL, "create table mod("                          );
    pchUsesSQL += sprintf( pchUsesSQL, "pky integer "                               );
    pchUsesSQL += sprintf( pchUsesSQL, "not null on conflict fail "                 );
    pchUsesSQL += sprintf( pchUsesSQL, "primary key on conflict fail "              );
    pchUsesSQL += sprintf( pchUsesSQL, "autoincrement unique on conflict fail,"     );
    pchUsesSQL += sprintf( pchUsesSQL, "fac integer not null on conflict fail,"     );
    pchUsesSQL += sprintf( pchUsesSQL, "dev integer not null on conflict fail,"     );
    pchUsesSQL += sprintf( pchUsesSQL, "ddvsv integer not null on conflict fail,"   );
    pchUsesSQL += sprintf( pchUsesSQL, "ddvtm integer not null on conflict fail);" );

    pchUsesSQL    = achUsesSQL;
    insApiReturn = sqlite3_exec( dbsUsageIN, pchUsesSQL, NULL, 0, &pchErrorIN );
    if( insApiReturn != SQLITE_OK )
    {
        sqlite3_free( pchErrorIN );
        pchErrorIN = NULL;
        return( false );
    };
    return( true );
}

/**
*************************************************************************
* Class public function.
*************************************************************************
*/
bool CNotifyLog::InsertMODsTableSQL( int useDevice )
```

```
<terminated> usenotify [C/C++ Local Application] /home/nkota/workspace/projects/usenotify/Debug/usenotify (09/11/2008 4:31 PM)
fac = 2
dev = 1001
ddvsv = 2
ddvtm = 1221168700
```

4.2.3 Screen Shot of the Display Data Telnet

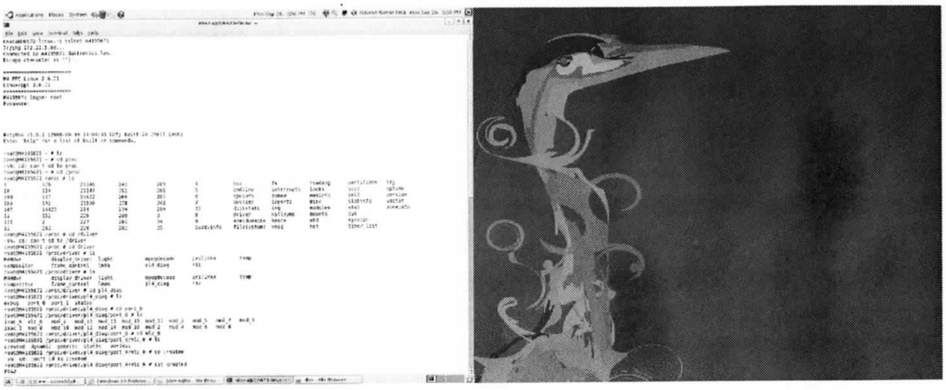
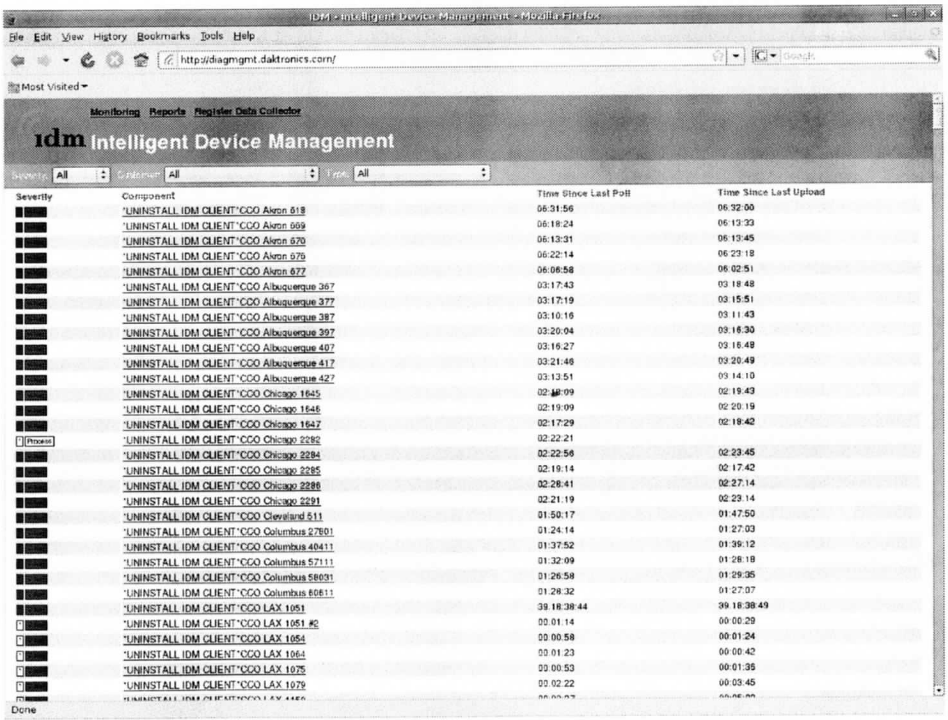


Figure 16: Screen Shot of the Display Data Telnet

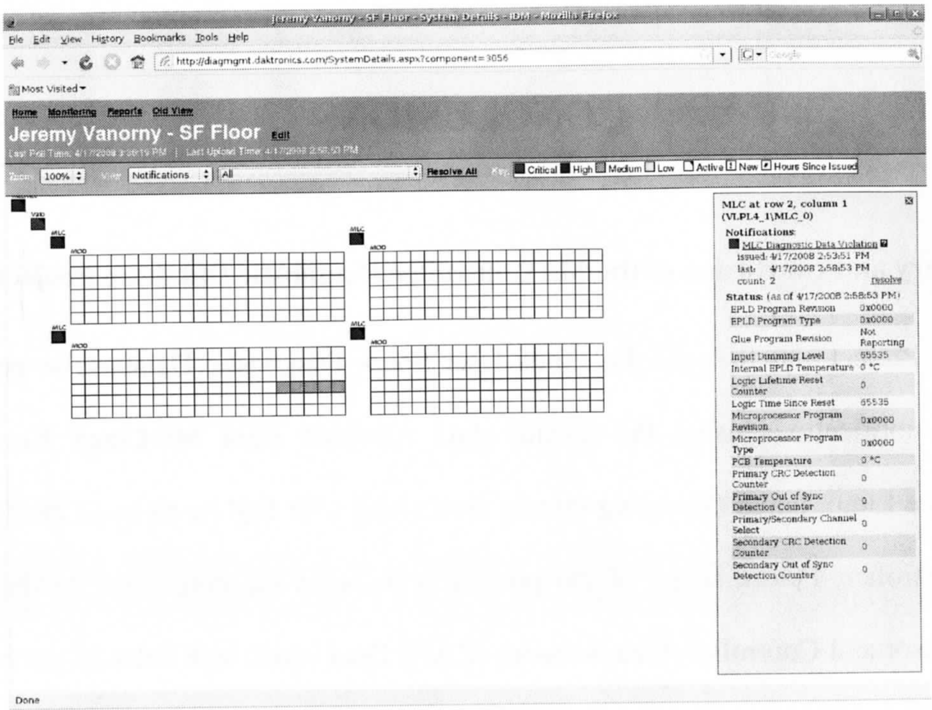
4.2.4 Screen Shot of the IDM at server level


IDM - Intelligent Device Management - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://diagmgmt.daktronics.com/
Google
Most Visited
Monitoring Reports Register Data Collector
idm Intelligent Device Management
Severity: All
Customer: All
Type: All
Done

Severity	Component	Time Since Last Poll	Time Since Last Upload
■	'UNINSTALL IDM CLIENT'CCO Akron 618	06:31:56	06:32:00
■	'UNINSTALL IDM CLIENT'CCO Akron 669	06:18:24	06:13:33
■	'UNINSTALL IDM CLIENT'CCO Akron 670	06:13:31	06:13:45
■	'UNINSTALL IDM CLIENT'CCO Akron 676	06:22:14	06:23:18
■	'UNINSTALL IDM CLIENT'CCO Akron 677	06:06:58	06:02:51
■	'UNINSTALL IDM CLIENT'CCO Albuquerque 367	03:17:43	03:18:48
■	'UNINSTALL IDM CLIENT'CCO Albuquerque 377	03:17:19	03:15:51
■	'UNINSTALL IDM CLIENT'CCO Albuquerque 387	03:10:16	03:11:43
■	'UNINSTALL IDM CLIENT'CCO Albuquerque 397	03:20:04	03:16:30
■	'UNINSTALL IDM CLIENT'CCO Albuquerque 407	03:16:27	03:16:48
■	'UNINSTALL IDM CLIENT'CCO Albuquerque 417	03:21:46	03:20:49
■	'UNINSTALL IDM CLIENT'CCO Albuquerque 427	03:13:51	03:14:10
■	'UNINSTALL IDM CLIENT'CCO Chicago 1645	02:18:09	02:19:43
■	'UNINSTALL IDM CLIENT'CCO Chicago 1646	02:19:09	02:20:19
■	'UNINSTALL IDM CLIENT'CCO Chicago 1647	02:17:29	02:18:42
□ Process	'UNINSTALL IDM CLIENT'CCO Chicago 2282	02:22:21	
■	'UNINSTALL IDM CLIENT'CCO Chicago 2284	02:22:56	02:23:45
■	'UNINSTALL IDM CLIENT'CCO Chicago 2285	02:19:14	02:17:42
■	'UNINSTALL IDM CLIENT'CCO Chicago 2286	02:28:41	02:27:14
■	'UNINSTALL IDM CLIENT'CCO Chicago 2291	02:21:19	02:23:14
■	'UNINSTALL IDM CLIENT'CCO Cleveland 511	01:50:17	01:47:50
■	'UNINSTALL IDM CLIENT'CCO Columbus 27601	01:24:14	01:27:03
■	'UNINSTALL IDM CLIENT'CCO Columbus 40411	01:37:52	01:39:12
■	'UNINSTALL IDM CLIENT'CCO Columbus 57111	01:32:09	01:28:18
■	'UNINSTALL IDM CLIENT'CCO Columbus 58031	01:26:58	01:29:35
■	'UNINSTALL IDM CLIENT'CCO Columbus 60611	01:28:32	01:27:07
■	'UNINSTALL IDM CLIENT'CCO LAX 1051	39.18:38:44	39.18:38:49
□ ■	'UNINSTALL IDM CLIENT'CCO LAX 1051 #2	00:01:14	00:00:29
□ ■	'UNINSTALL IDM CLIENT'CCO LAX 1054	00:00:58	00:01:24
□ ■	'UNINSTALL IDM CLIENT'CCO LAX 1064	00:01:23	00:00:42
□ ■	'UNINSTALL IDM CLIENT'CCO LAX 1075	00:00:53	00:01:36
□ ■	'UNINSTALL IDM CLIENT'CCO LAX 1079	00:02:22	00:03:45

Figure 17: Screen Shot of the IDM at server level

4.2.5 Screen Shot of the System Details


Jeremy Vanorny - SF Floor - System Details - IDM - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://diagmgmt.daktronics.com/SystemDetails.aspx?component=3056
Most Visited
Home Monitoring Reports Old View
Jeremy Vanorny - SF Floor Edit
Last Poll Time: 4/17/2008 3:00:19 PM | Last Upload Time: 4/17/2008 2:58:53 PM
Zoom 100% View Notifications All Resolve All
Key: Critical High Medium Low Active New Hours Since Issued
VALO
MLC
MOD
Done

MLC at row 2, column 1
(VLPL4_1\MLC_0)

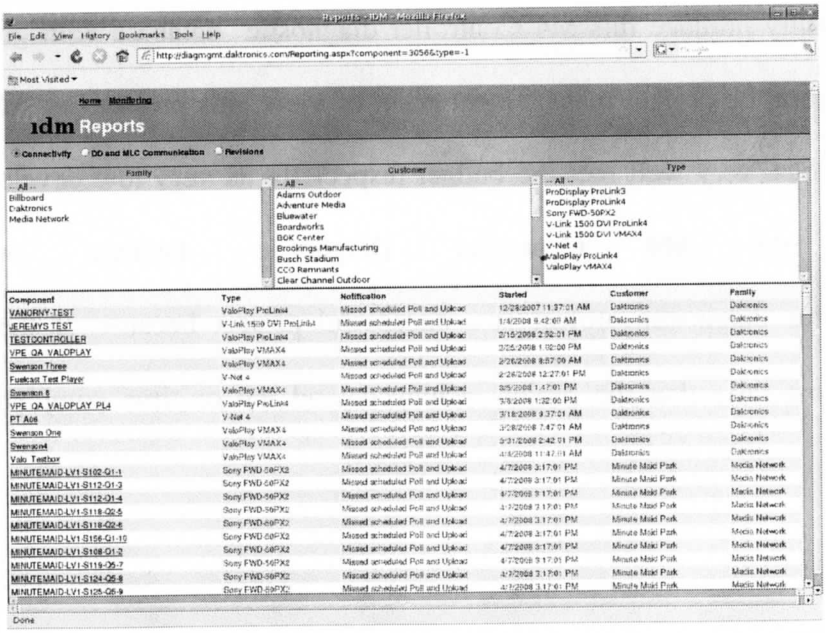
Notifications:
MLC Diagnostic Data Violation
issued: 4/17/2008 2:53:51 PM
last: 4/17/2008 2:58:53 PM
count: 2 resolve

Status: (as of 4/17/2008 2:58:53 PM)

EPLD Program Revision	0x0000
EPLD Program Type	0x0000
Glue Program Revision	Not Reporting
Input Dimming Level	65535
Internal EPLD Temperature	0 °C
Logic Lifetime Reset Counter	0
Logic Time Since Reset	65535
Microprocessor Program Revision	0x0000
Microprocessor Program Type	0x0000
PCB Temperature	0 °C
Primary CRC Detection Counter	0
Primary Out of Sync Detection Counter	0
Primary/Secondary Channel Select	0
Secondary CRC Detection Counter	0
Secondary Out of Sync Detection Counter	0

Figure 18: Screen Shot of System Details

4.2.6 Screen Shot of the Reports


Reports - IDM - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://diagmgmt.daktronics.com/Reporting.aspx?component=3056&type=-1
Most Visited
Home Monitoring
idm Reports
Done

● Connectivity ○ DD and MLC Communication ○ Revisions

Family	Customer	Type
-- All -- Billboard Daktronics Media Network	-- All -- Adams Outdoor Adventure Media Bluewater Boardworks BOK Center Brookings Manufacturing Busch Stadium CCO Remnants Clear Channel Outdoor	-- All -- ProDisplay ProLink3 ProDisplay ProLink4 Sony FWD-50PX2 V-Link 1500 DVI ProLink4 V-Link 1500 DVI VMAX4 V-Net 4 ValoPlay ProLink4 ValoPlay VMAX4

Component	Type	Notification	Started	Customer	Family
VANORNY-TEST	ValoPlay ProLink4	Missed scheduled Poll and Upload	12/28/2007 11:37:01 AM	Daktronics	Daktronics
JEREMYS TEST	V-Link 1500 DVI ProLink4	Missed scheduled Poll and Upload	1/4/2008 9:42:00 AM	Daktronics	Daktronics
TESTCONTROLLER	ValoPlay ProLink4	Missed scheduled Poll and Upload	2/15/2008 2:52:01 PM	Daktronics	Daktronics
VPE_QA_VALOPLAY	ValoPlay VMAX4	Missed scheduled Poll and Upload	2/25/2008 1:02:00 PM	Daktronics	Daktronics
Swenson Three	ValoPlay VMAX4	Missed scheduled Poll and Upload	2/26/2008 8:57:00 AM	Daktronics	Daktronics
Fuelcast Test Player	V-Net 4	Missed scheduled Poll and Upload	2/26/2008 12:27:01 PM	Daktronics	Daktronics
Swenson 6	ValoPlay VMAX4	Missed scheduled Poll and Upload	3/5/2008 1:47:01 PM	Daktronics	Daktronics
VPE_QA_VALOPLAY_PL4	ValoPlay ProLink4	Missed scheduled Poll and Upload	3/6/2008 1:32:00 PM	Daktronics	Daktronics
PT A06	V-Net 4	Missed scheduled Poll and Upload	3/18/2008 9:37:01 AM	Daktronics	Daktronics
Swenson One	ValoPlay VMAX4	Missed scheduled Poll and Upload	3/28/2008 7:47:01 AM	Daktronics	Daktronics
Swenson4	ValoPlay VMAX4	Missed scheduled Poll and Upload	3/31/2008 2:42:01 PM	Daktronics	Daktronics
Valo_Testbox	ValoPlay VMAX4	Missed scheduled Poll and Upload	4/4/2008 11:47:01 AM	Daktronics	Daktronics
MINUTEMAID-LV1-S102-O1-1	Sony FWD-50PX2	Missed scheduled Poll and Upload	4/7/2008 3:17:01 PM	Minute Maid Park	Media Network
MINUTEMAID-LV1-S112-O1-3	Sony FWD-50PX2	Missed scheduled Poll and Upload	4/7/2008 3:17:01 PM	Minute Maid Park	Media Network
MINUTEMAID-LV1-S112-O1-4	Sony FWD-50PX2	Missed scheduled Poll and Upload	4/7/2008 3:17:01 PM	Minute Maid Park	Media Network
MINUTEMAID-LV1-S118-O2-5	Sony FWD-50PX2	Missed scheduled Poll and Upload	4/7/2008 3:17:01 PM	Minute Maid Park	Media Network
MINUTEMAID-LV1-S118-O2-6	Sony FWD-50PX2	Missed scheduled Poll and Upload	4/7/2008 3:17:01 PM	Minute Maid Park	Media Network
MINUTEMAID-LV1-S156-O1-10	Sony FWD-50PX2	Missed scheduled Poll and Upload	4/7/2008 3:17:01 PM	Minute Maid Park	Media Network
MINUTEMAID-LV1-S108-O1-2	Sony FWD-50PX2	Missed scheduled Poll and Upload	4/7/2008 3:17:01 PM	Minute Maid Park	Media Network
MINUTEMAID-LV1-S119-O5-7	Sony FWD-50PX2	Missed scheduled Poll and Upload	4/7/2008 3:17:01 PM	Minute Maid Park	Media Network
MINUTEMAID-LV1-S124-O5-8	Sony FWD-50PX2	Missed scheduled Poll and Upload	4/7/2008 3:17:01 PM	Minute Maid Park	Media Network
MINUTEMAID-LV1-S125-O5-9	Sony FWD-50PX2	Missed scheduled Poll and Upload	4/7/2008 3:17:01 PM	Minute Maid Park	Media Network

Figure 19: Screen Shot of the Reports

CHAPTER 5

CONCLUSIONS

5.0 Summary and conclusion of the M4 Controller Diagnostic Database Project

The Project was to build the Linux-User Space Diagnostic Database to generate the Diagnostic notifications using the display data provided from M4-Linux Kernel space. Ubuntu is the Linux-based Operating system used along with SQLite to build the database on the M4 controller. Future scope of the project is to build the diagnostic database for the display, Sensor and Controller Area Network (CAN) Data which will delivers the notification to Daktronics technicians from the Database which is built on the M4 controller.

This project combines the current technologies and the various knowledge that are learned from the Dakota State University, Master of Science Information Systems degree courses, to successfully produce this M4 Controller diagnostic database with good quality and at low cost. From the project management point of view, this project has delivered the required functionalities, very well met the budget (especially its very low developing cost) and delivered the M4 Controller Diagnostic database on time.

Therefore, this diagnostic project is a successful project because of the excellent performance-over-cost ratio that the project achieved, and the on-time completion of the project.

REFERENCES

Daktronics, Inc (2009, February 16).*M4 Controller*. Retrieved September 5,2008 from South Dakota, Commercial Web site: http://www.daktronics.com/Pages/default.aspx

D. Richard Hipp (2009, February 16). *SQLite*. Retrieved October18, 2008 from Hwaci - Applied Software Research, Web site: http://www.sqlite.org/

Vijay Mathew Pandyalakal (2007,January 14).*A C++ wrapper to SQLite database*. Retrieved November 5,2008 from ,database Web site: http://www.codeproject.com/KB/database/vmembedded_db_cpp.aspx

Rob Groves (2004,october 28).*CppSQLite*. Retrieved September 25,2008 from United Kingdom, database Web site http://www.codeproject.com/KB/database/CppSQLite.aspx

Free Software Foundation, Inc (2006,July 20).*SQLite library on Linux*. Retrieved October 15,2008 from , SQLitePass Database Objects for Lazarus Website:http://source.online.free.fr/Linux_HowToCompileSQLite.html

Alan Pope (2008,August 5). *Installation*. Retrieved July 1,2008 from Ubuntu Development Team, SQLitePass Database Objects for Lazarus Web site: https://help.ubuntu.com/community/Installation

APPENDICES

APPENDIX A: Users' manual

Code redacted for privacy.