ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2014

Remote Monitoring and Control by Embedded Database Design and Web Server Implementation

C.Balaji Kumar[1], C.Madhu[1], K.Tirumala Reddy[3]

Assistant Professor, Department of ECE, S V College of Engineering, Tirupati, A.P, India[1,2,3]

ABSTRACT: This paper describes the design of embedded system that enables the remote access of database by the implementation of web server. Porting of Linux to ARM9 is carried out to enable further porting of embedded webserver (BOA server) and SQLite database. The ported database is implemented by creating a table, storing the data into the database and retrieving the same. An application is developed for displaying the data entered into the database in a specific format. BOA web server is ported to home gateway platform and implemented in order to enable further enhancements such as remote monitoring of data stored in the database. The web server is implemented by displaying the web page stored in the server, when the concerned server address is entered into the browser. The SQLite database along with BOA web server on ARM platform can be used in industries, remote areas, even at homes for monitoring and controlling the status of appliances and machinery, by adding additional enhancements and doing slight modifications according to the application.

KEYWORDS: SQLite, Database, Web server, BOA, Remote access and control

I. INTRODUCTION

The need of remote monitoring and access for various embedded applications has increased the demand for investigating an effective technique in terms of cost as well as power. Various remote monitoring and controlling techniques are studied [3]-[6], and it is identified that the best results can be obtained, when the database and web server are designed specifically for embedded applications [1]. SQLite database and Boa webserver are such softwares [1], [4], which satisfy the requirements of all embedded applications. An attempt of making use of the best features of both SQLite database and Boa webserver has been proposed.

The block diagram of the proposed remote monitoring system is as shown in Fig1.1. MINI2440 development board is used in the system design.The linux operating system (linux-2.6.32.2) is ported to ARM9 platform. Both the SQLite database and boa webserver are ported to linux platform on MINI2440 development board. The version of SQLite database, used in the proposal is SQLite-3.6.22 and the version of boa web server is boa-0.94.13. The SQLite database is implemented by entering, retrieving and modifying the data using SQLite queries and commands. The ARM development board itself acts as server. When the client requests for the data, stored in the server by specifying appropriate address in the browser, the server sends it using http protocol. Thus, webserver is also implemented.

As linux is a free open-source operating system and has to be customized and compiled for every new CPU architecture, it is necessary to make the cross-compiler (such as gnu cross compiler) accessible on the execution path in order to build the linux kernel. As the kernel contains lots of device drivers, network protocols, file system drivers etc., it is to be configured as per the requirement.

SQLite database, which is an in-process library has many merits such as server less, zero configuration etc., makes it suitable especially for embedded applications. The code for SQLite is also an open source and it has cross-platform file format

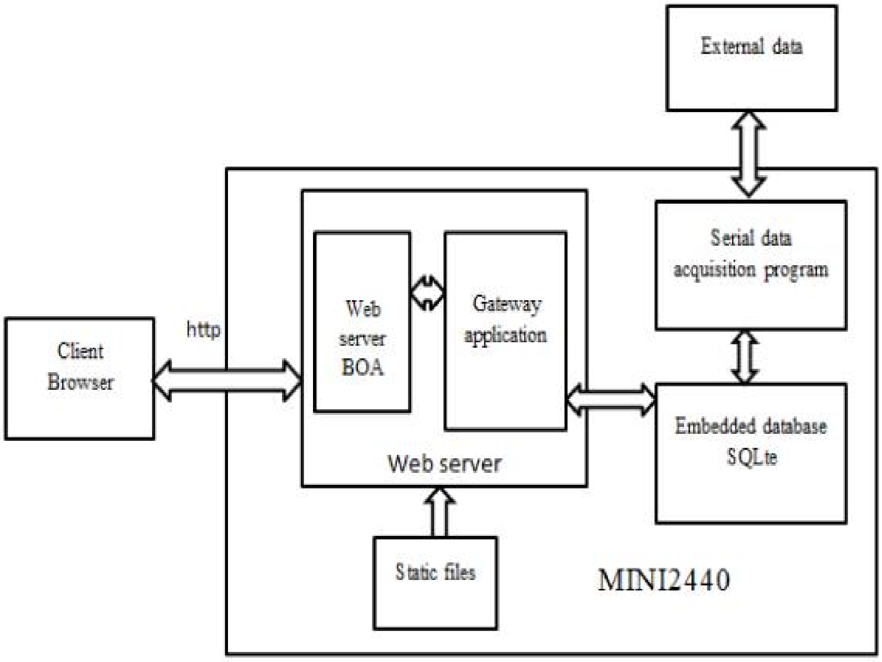


Fig. 1: Block Diagram of the Project

The features such as small-footprint web server, open source make Boa suitable for embedded applications. In addition to this, it has many more desirable features such as single-tasking, high processing speed, automatic directory generation, automatic file extraction etc.

II. PORTING OF LINUX KERNEL TO ARM9 PLATFORM

In order to get the kernel compiled, the kernel source code, build tools, kernel configuration file are needed. It is also required to have root or sudo privileges for the final stages of the process. For compiling the kernel on the ARM9 board, cross compiler environment should be created.

A. ESTABLISHING CROSS COMPILER ENVIRONMENT:

- The source code tar file of arm gcc cross compiler is copied to a directory such as (/opt/Friendlyarm/micro2440) and thethe tar file is extracted.
- The compiler path to the system environmentvariables is added.
- The path in /root/.bashrc file is set.
- For knowing whether the cross compiler has been installed or not, the following command is used:

```
# arm -linux -gcc -v
```

B. COMPILING THE KERNEL:

- The tar file is extracted in the working directory.
- To make the ARM platform as default target platform for linux, Makefile is edited appropriately.
- The default kernel configuration file is used, for testing linux compilation:

```
# make S3C2410_defconfig
# make
```

- To determine the target platform,

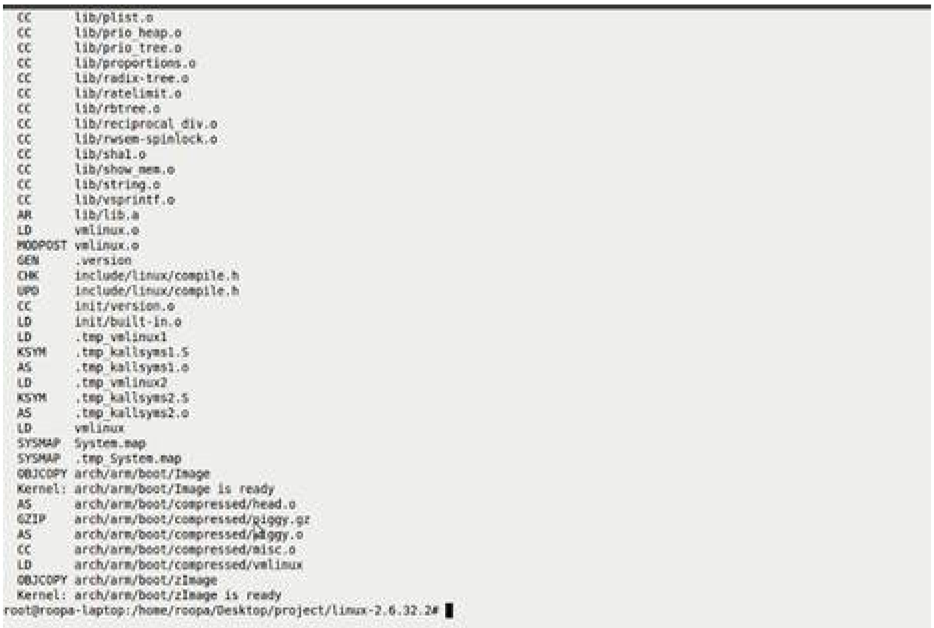
```
# gedit arch/arm/tools/mach_types
```

DOI:10.15662/IJAREEIE.2014.0302057

#gedit arch/arm/mach_S3C2440/mach_smdk2440.c

- To test the compilation,

# make mini2440_defconfig
# make zImage



```
  CC      lib/plist.o
  CC      lib/prio_heap.o
  CC      lib/prio_tree.o
  CC      lib/proportions.o
  CC      lib/radix-tree.o
  CC      lib/ratelimit.o
  CC      lib/rbtree.o
  CC      lib/reciprocal_div.o
  CC      lib/rwsem-spinlock.o
  CC      lib/sha1.o
  CC      lib/show_mem.o
  CC      lib/string.o
  CC      lib/vsprintf.o
  AR      lib/lib.a
  LD      vmlinux.o
  MODPOST vmlinux.o
  GEN     .version
  CHK     include/linux/compile.h
  UPD     include/linux/compile.h
  CC      init/version.o
  LD      init/built-in.o
  LD      .tmp_vmlinux1
  KSYM    .tmp_kallsyms1.S
  AS      .tmp_kallsyms1.o
  LD      .tmp_vmlinux2
  KSYM    .tmp_kallsyms2.S
  AS      .tmp_kallsyms2.o
  LD      vmlinux
  SYSMAP  System.map
  SYSMAP  .tmp_System.map
  OBJCOPY arch/arm/boot/Image
  Kernel: arch/arm/boot/Image is ready
  AS      arch/arm/boot/compressed/head.o
  GZIP    arch/arm/boot/compressed/piggy.gz
  AS      arch/arm/boot/compressed/piggy.o
  CC      arch/arm/boot/compressed/misc.o
  LD      arch/arm/boot/compressed/vmlinux
  OBJCOPY arch/arm/boot/zImage
  Kernel: arch/arm/boot/zImage is ready
root@roopa-laptop:/home/roopa/Desktop/project/linux-2.6.32.2#
```

Fig. 2.1: Compiling zImage

- For configuring Kernel menu,

# make menuconfig
# gedit arch/arm/mach-S3C2440/Kconfig
# gedit arch/arm/mach-S3C2440/Makefile

- For compiling uImage or zImage respectively,

# make uImage or #make zImage

C. BUSYBOX:

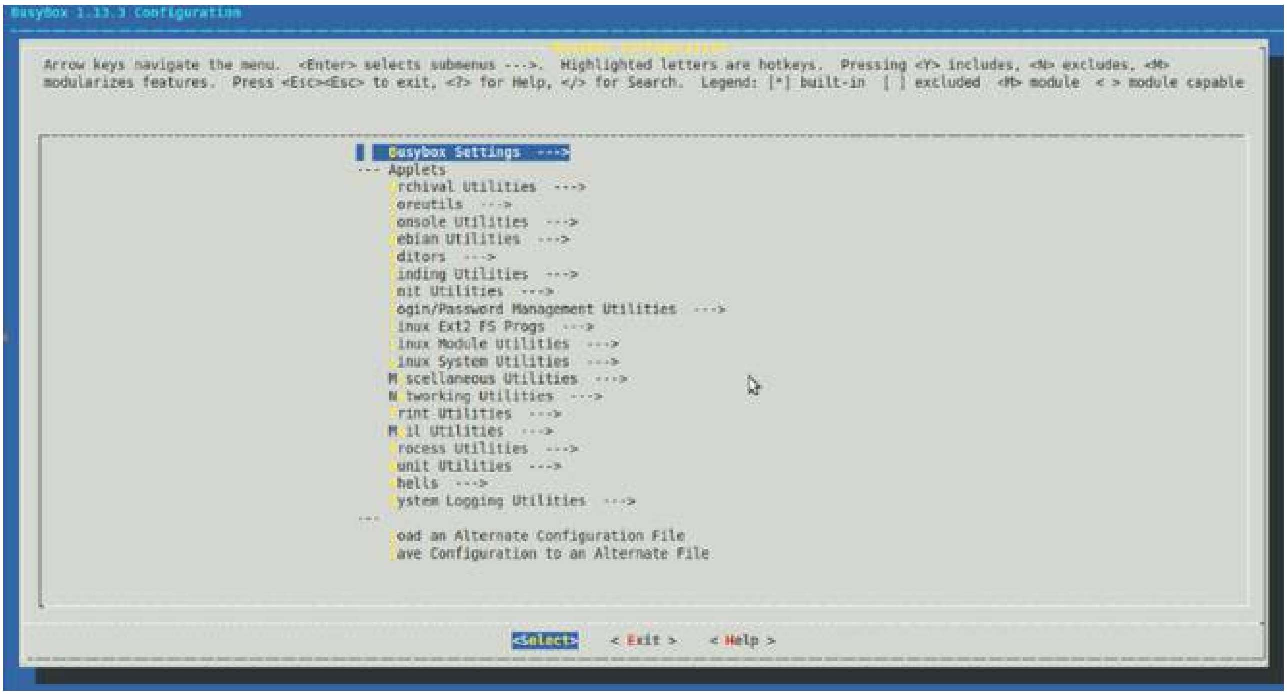
Busybox is a package that provides all the basic things required for a root file system in a very compact form. It is remarkably easy to configure, compile, and use, and it has the potential to significantly reduce the overall system resources required to support a wide collection of common Linux utilities. Busybox installation steps:

- Busybox source code is downloaded into testing directory and extract the file.
- Busybox is configured using 'make menuconfig' instruction like kernel does.

# make menuconfig

- Busybox settings – installation options – (./install) Busybox installation prefix – path of testing directory

DOI:10.15662/IJAREEIE.2014.0302057



```
Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable
```

Fig. 2.2: Configuring busybox menu

• The busyboxis compiled using
# make
# make install

• The following steps are, to check the libraries needed by busybox to run. So move into the bin directory, where the busybox resides and with the readelf command, it can be checked, whether the required libraries are there or not.

```
# $CROSS COMPILE “read elf -a busybox|grep lib
# cd ..
# mkdir lib
# cd lib
```

• The appropriate shared libraries are copied into the library directory.

```
#cpusr/local/arm/4.3.2/arm-none-linuxgnueabi/
libc/armv4t/lib/ld-linux.so.3
#cpusr/local/arm/4.3.2/arm-none-linuxgnueabi/
libc/armv4t/lib/libm.so.6
#cpusr/local/arm/4.3.2/arm-none-linuxgnueabi/
libc/armv4t/lib/libc.so.6
```

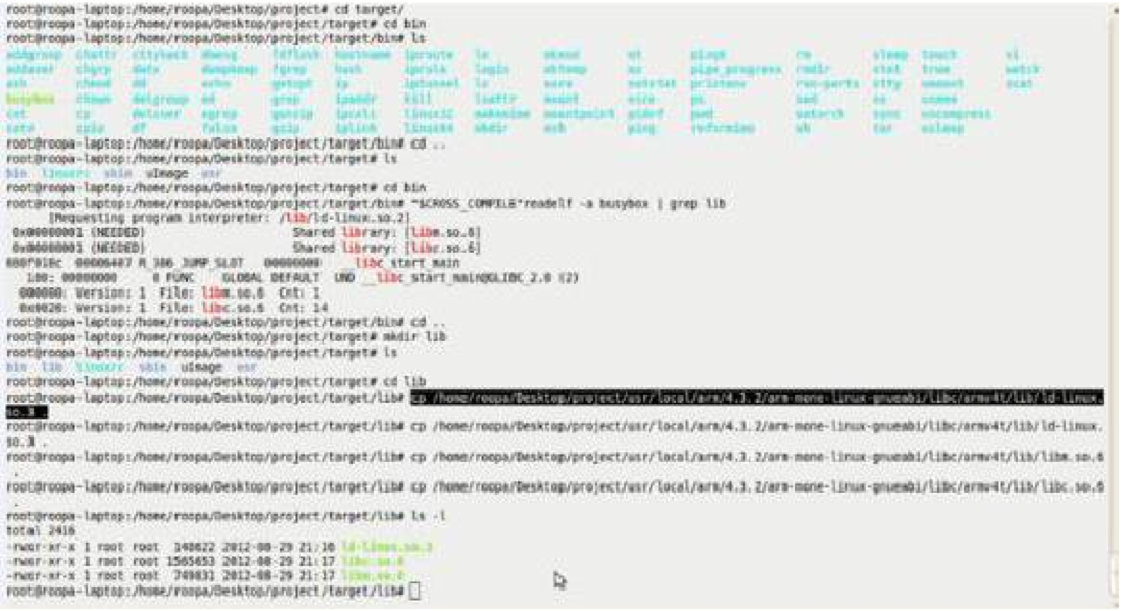
DOI:10.15662/IJAREEIE.2014.0302057

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2014



```
root@roopa-laptop:/home/roopa/Desktop/project# cd target/
root@roopa-laptop:/home/roopa/Desktop/project/target# cd bin
root@roopa-laptop:/home/roopa/Desktop/project/target/bin# ls
addgroup  chattr  cttyhack  dmesg          fdflush  hostname  iproute   ln        mknod       mt       ping6          rm         sleep  touch       vi
adduser   chgrp   date      dnsdomainname  fgrep    hush      iprule    login     mktemp      mv       pipe_progress  rmdir      stat   true        watch
ash       chmod   dd        echo           getopt   ip        iptunnel  ls        more        netstat  printenv       run-parts  stty   umount      zcat
busybox   chown   delgroup  ed             grep     ipaddr    kill      lsattr    mount       nice     ps             sed        su     uname
cat       cp      deluser   egrep          gunzip   ipcalc    linux32   makemime  mountpoint  pidof    pwd            setarch    sync   uncompress
catv      cpio    df        false          gzip     iplink    linux64   mkdir     msh         ping     reformime      sh         tar    usleep
root@roopa-laptop:/home/roopa/Desktop/project/target/bin# cd ..
root@roopa-laptop:/home/roopa/Desktop/project/target# ls
bin  linuxrc  sbin  uImage  usr
root@roopa-laptop:/home/roopa/Desktop/project/target# cd bin
root@roopa-laptop:/home/roopa/Desktop/project/target/bin# "$CROSS_COMPILE"readelf -a busybox | grep lib
      [Requesting program interpreter: /lib/ld-linux.so.2]
 0x00000001 (NEEDED)                     Shared library: [libm.so.6]
 0x00000001 (NEEDED)                     Shared library: [libc.so.6]
000f01bc  00000407 R_386_JUMP_SLOT   00000000   __libc_start_main
   180: 00000000     0 FUNC    GLOBAL DEFAULT  UND __libc_start_main@GLIBC_2.0 (2)
  000000: Version: 1  File: libm.so.6  Cnt: 1
  0x0020: Version: 1  File: libc.so.6  Cnt: 14
root@roopa-laptop:/home/roopa/Desktop/project/target/bin# cd ..
root@roopa-laptop:/home/roopa/Desktop/project/target# mkdir lib
root@roopa-laptop:/home/roopa/Desktop/project/target# ls
bin  lib  linuxrc  sbin  uImage  usr
root@roopa-laptop:/home/roopa/Desktop/project/target# cd lib
root@roopa-laptop:/home/roopa/Desktop/project/target/lib# cp /home/roopa/Desktop/project/usr/local/arm/4.3.2/arm-none-linux-gnueabi/libc/armv4t/lib/ld-linux.
so.3 .
root@roopa-laptop:/home/roopa/Desktop/project/target/lib# cp /home/roopa/Desktop/project/usr/local/arm/4.3.2/arm-none-linux-gnueabi/libc/armv4t/lib/ld-linux.
so.3 .
root@roopa-laptop:/home/roopa/Desktop/project/target/lib# cp /home/roopa/Desktop/project/usr/local/arm/4.3.2/arm-none-linux-gnueabi/libc/armv4t/lib/libm.so.6
 .
root@roopa-laptop:/home/roopa/Desktop/project/target/lib# cp /home/roopa/Desktop/project/usr/local/arm/4.3.2/arm-none-linux-gnueabi/libc/armv4t/lib/libc.so.6
 .
root@roopa-laptop:/home/roopa/Desktop/project/target/lib# ls -l
total 2416
-rwxr-xr-x 1 root root  140622 2012-08-29 21:16 ld-linux.so.3
-rwxr-xr-x 1 root root 1565653 2012-08-29 21:17 libc.so.6
-rwxr-xr-x 1 root root  749831 2012-08-29 21:17 libm.so.6
root@roopa-laptop:/home/roopa/Desktop/project/target/lib# 
```

Fig. 2.3: Copying the required libraries into target directory

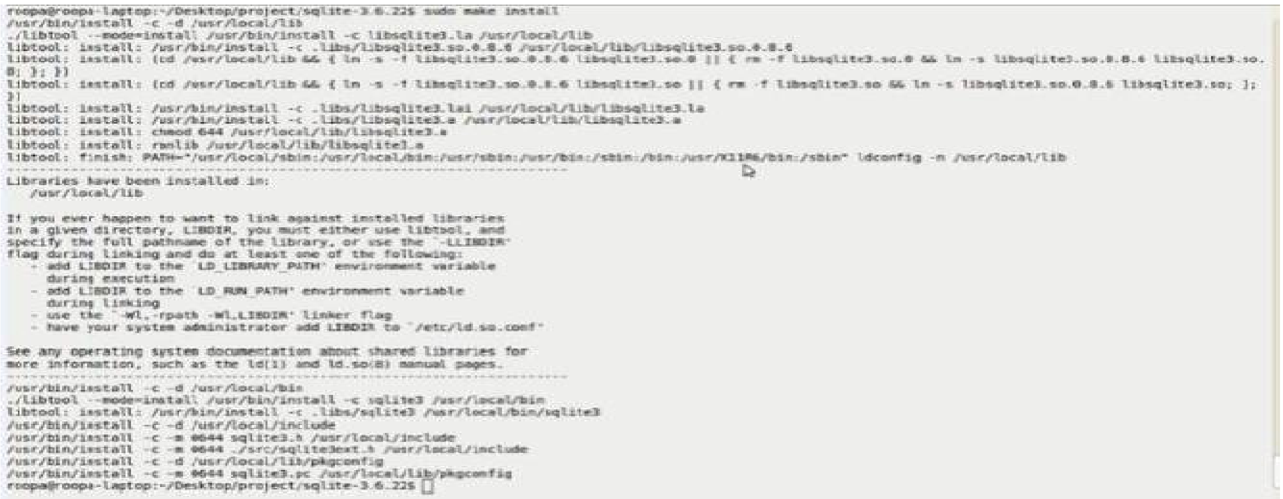
III. PORTING OF SQLITE TO THE DEVELOPMENT BOARD

- The sqlite source code is downloaded and extracted.
- A directory build is created as follows:

# mkdir build
# cd build

- Sqlite-3.6.22 build directory configure script generates a Makefile file:

./configure -host = arm-linux -prefix = <path>/sqlite- 3.6.22/build/target
# make
# make install



```
roopa@roopa-laptop:~/Desktop/project/sqlite-3.6.22$ sudo make install
/usr/bin/install -c -d /usr/local/lib
./libtool --mode=install /usr/bin/install -c libsqlite3.la /usr/local/lib
libtool: install: /usr/bin/install -c .libs/libsqlite3.so.0.8.6 /usr/local/lib/libsqlite3.so.0.8.6
libtool: install: (cd /usr/local/lib && { ln -s -f libsqlite3.so.0.8.6 libsqlite3.so.0 || { rm -f libsqlite3.so.0 && ln -s libsqlite3.so.0.8.6 libsqlite3.so.
0; }; })
libtool: install: (cd /usr/local/lib && { ln -s -f libsqlite3.so.0.8.6 libsqlite3.so || { rm -f libsqlite3.so && ln -s libsqlite3.so.0.8.6 libsqlite3.so; };
})
libtool: install: /usr/bin/install -c .libs/libsqlite3.lai /usr/local/lib/libsqlite3.la
libtool: install: /usr/bin/install -c .libs/libsqlite3.a /usr/local/lib/libsqlite3.a
libtool: install: chmod 644 /usr/local/lib/libsqlite3.a
libtool: install: ranlib /usr/local/lib/libsqlite3.a
libtool: finish: PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/X11R6/bin:/sbin" ldconfig -n /usr/local/lib
----------------------------------------------------------------------
Libraries have been installed in:
   /usr/local/lib

If you ever happen to want to link against installed libraries
in a given directory, LIBDIR, you must either use libtool, and
specify the full pathname of the library, or use the `-LLIBDIR'
flag during linking and do at least one of the following:
   - add LIBDIR to the `LD_LIBRARY_PATH' environment variable
     during execution
   - add LIBDIR to the `LD_RUN_PATH' environment variable
     during linking
   - use the `-Wl,-rpath -Wl,LIBDIR' linker flag
   - have your system administrator add LIBDIR to `/etc/ld.so.conf'

See any operating system documentation about shared libraries for
more information, such as the ld(1) and ld.so(8) manual pages.
----------------------------------------------------------------------
/usr/bin/install -c -d /usr/local/bin
./libtool --mode=install /usr/bin/install -c sqlite3 /usr/local/bin
libtool: install: /usr/bin/install -c .libs/sqlite3 /usr/local/bin/sqlite3
/usr/bin/install -c -d /usr/local/include
/usr/bin/install -c -m 0644 sqlite3.h /usr/local/include
/usr/bin/install -c -m 0644 ./src/sqlite3ext.h /usr/local/include
/usr/bin/install -c -d /usr/local/lib/pkgconfig
/usr/bin/install -c -m 0644 sqlite3.pc /usr/local/lib/pkgconfig
roopa@roopa-laptop:~/Desktop/project/sqlite-3.6.22$ 
```

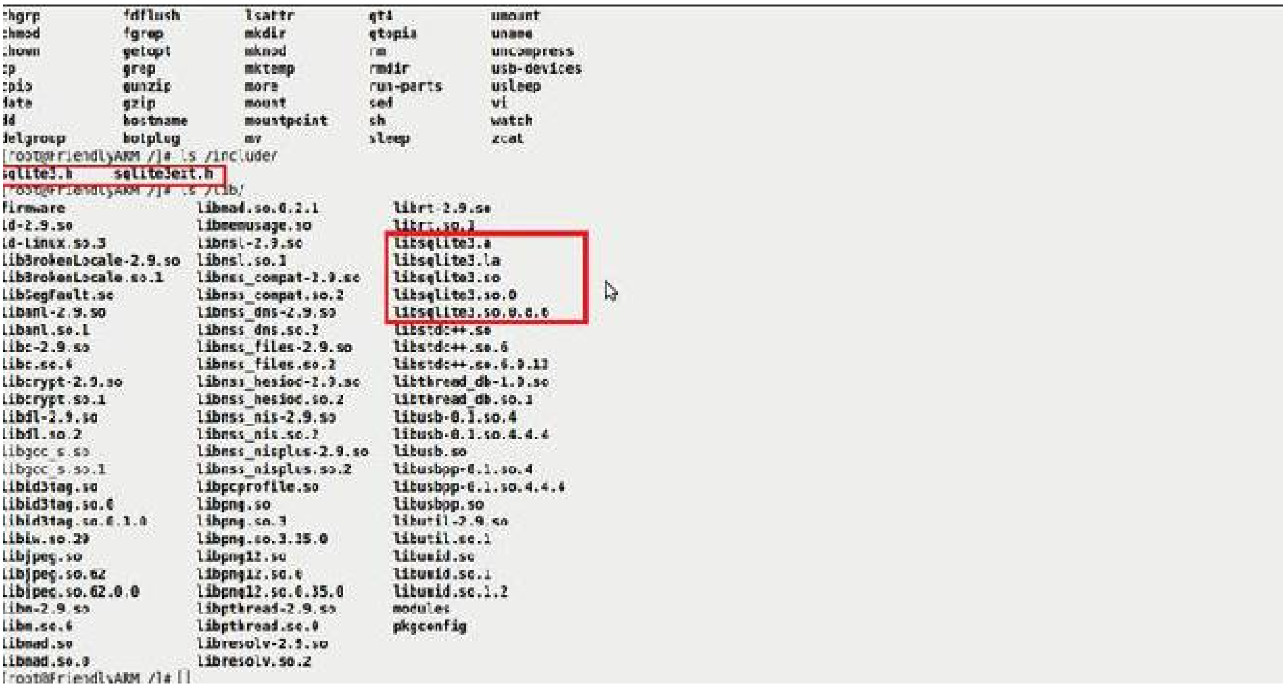
Fig. 3.1: Installation of SQLite

DOI:10.15662/IJAREEIE.2014.0302057

- <path>/sqlite-3.6.22/build/target directory folder contains three main files, namely *bin*, *include*, *lib*.

- All the files under the *bin* file are downloaded to the */usr/local/bin* directory of development board. All the files under the *lib* are downloaded to the */usr/local/lib* directory of development board; and all the files of *include* directory which contains sqlite, C-language API header files used in the programming are downloaded to the /usr/local/include directory of development board.

- Soft link is formed between the library files of /usr/local/ and library files of development board as shown below:



```
chgrp          fdflush        lsattr         qt4            umount
chmod          fgrep          mkdir          qtopia         uname
chown          getopt         mknod          rm             uncompress
cp             grep           mktemp         rmdir          usb-devices
cpio           gunzip         more           run-parts      usleep
date           gzip           mount          sed            vi
dd             hostname       mountpoint     sh             watch
delgroup       hotplug        mv             sleep          zcat
[root@FriendlyARM /]# ls /include/
sqlite3.h      sqlite3ext.h
[root@FriendlyARM /]# ls /lib/
firmware                  libmad.so.0.2.1           librt-2.9.so
ld-2.9.so                 libmemusage.so            librt.so.1
ld-linux.so.3             libnsl-2.9.so             libsqlite3.a
libBrokenLocale-2.9.so    libnsl.so.1               libsqlite3.la
libBrokenLocale.so.1      libnss_compat-2.9.so      libsqlite3.so
libSegFault.so            libnss_compat.so.2        libsqlite3.so.0
libanl-2.9.so             libnss_dns-2.9.so         libsqlite3.so.0.8.6
libanl.so.1               libnss_dns.so.2           libstdc++.so
libc-2.9.so               libnss_files-2.9.so       libstdc++.so.6
libc.so.6                 libnss_files.so.2         libstdc++.so.6.0.13
libcrypt-2.9.so           libnss_hesiod-2.9.so      libthread_db-1.0.so
libcrypt.so.1             libnss_hesiod.so.2        libthread_db.so.1
libdl-2.9.so              libnss_nis-2.9.so         libusb-0.1.so.4
libdl.so.2                libnss_nis.so.2           libusb-0.1.so.4.4.4
libgcc_s.so               libnss_nisplus-2.9.so     libusb.so
libgcc_s.so.1             libnss_nisplus.so.2       libusbpp-0.1.so.4
libid3tag.so              libpcprofile.so           libusbpp-0.1.so.4.4.4
libid3tag.so.0            libpng.so                 libusbpp.so
libid3tag.so.0.3.0        libpng.so.3               libutil-2.9.so
libiw.so.29               libpng.so.3.35.0          libutil.so.1
libjpeg.so                libpng12.so               libuuid.so
libjpeg.so.62             libpng12.so.0             libuuid.so.1
libjpeg.so.62.0.0         libpng12.so.0.35.0        libuuid.so.1.2
libm-2.9.so               libpthread-2.9.so         modules
libm.so.6                 libpthread.so.0           pkgconfig
libmad.so                 libresolv-2.9.so
libmad.so.0               libresolv.so.2
[root@FriendlyARM /]# []
```

Fig. 3.2: SQLite library files are copied onto the board

The above step completes the porting of SQLite onto the development board.

IV. PORTING OF BOA TO THE DEVELOPMENT BOARD

- Boa source code is downloaded and decompressed into subdirectory of source code directory.

```
# tar -xzf boa-0.94.13.tar.gz
# cd boa-0.94.13/src
```

- Makefile is generated by

```
# ./configure
```

- Makefile is modified: Find CC=gcc and CPP=gcc –E, change them to the directories CC=/opt/host/bin/armlinux-gcc and CPP=opt/host/bin/arm-linux-gcc-E, in which the cross compiler is installed. Save and exit. Then run *make* command to compile to get the executable program boa.
- *Boa configuration:*

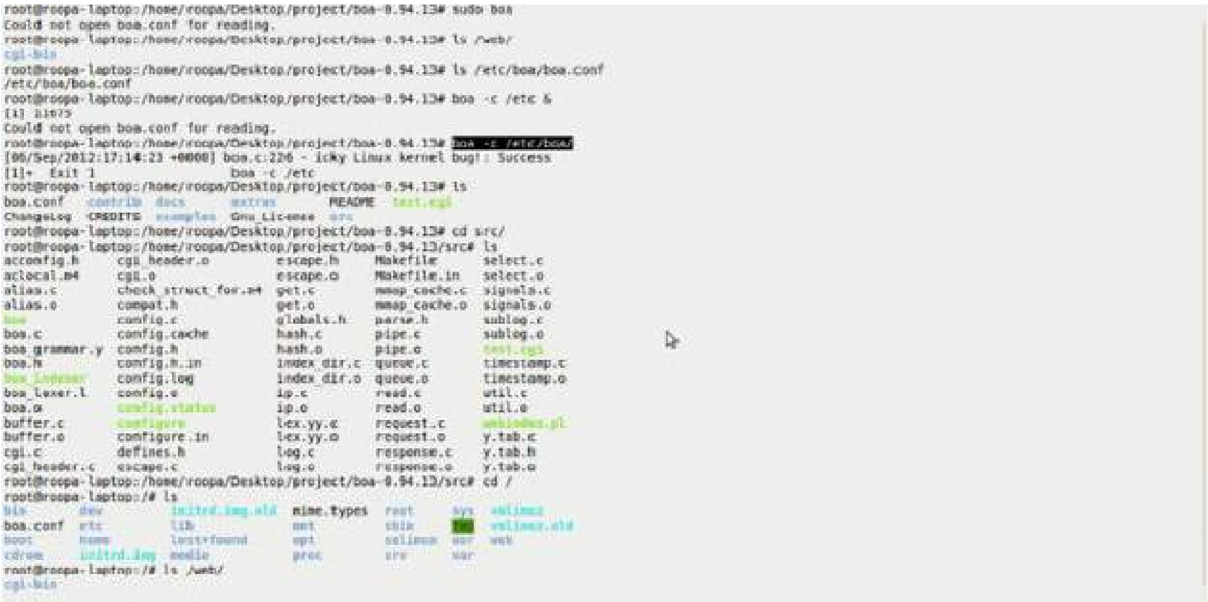
Boa needs to create a *boa* directory in */etc* directory on the virtual machine and put the main configuration file *boa.conf* into *boa*. In *boa* source directory, there has already an example boa.conf based on which, can be modified.

- *Group nogroup* is modified as *group 0*. Because there has no nogroup in file */etc/group*, so it is to be set as 0.

DOI:10.15662/IJAREEIE.2014.0302057

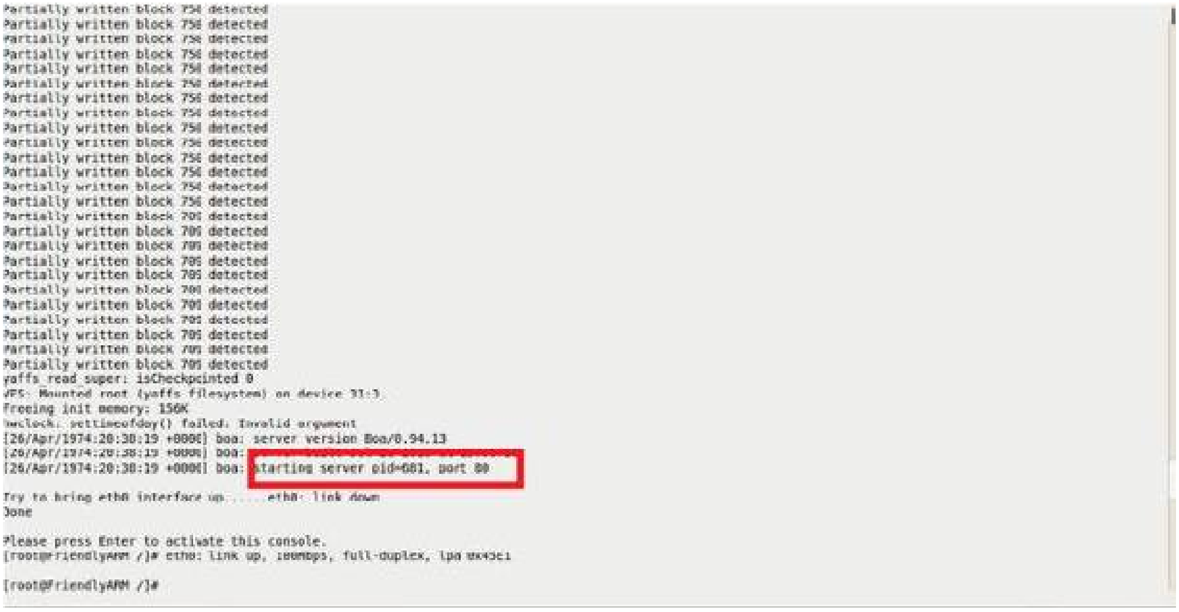
- There has *nobody* user in */etc/passwd*, so user nobody need not be modified.
- *ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/* is modified as *ScriptAlias /cgi-bin/ /var/www/cgi-bin/*, and others are default configurations.

• Log file is created in */var/log/boa*. The main directory for creating HTML documents is */var/www*, and stores static webpages into this directory. CGI scripts are generated in */var/www/cgi-bin*, and cgi scripts are stored in this directory. The file *mime.types* is copied to */etc* directory. Generally, it can be copied directly from */etc* directory of linux host.



```
root@roopa-laptop:/home/roopa/Desktop/project/boa-0.94.13# sudo boa
Could not open boa.conf for reading.
root@roopa-laptop:/home/roopa/Desktop/project/boa-0.94.13# ls /web/
cgi-bin
root@roopa-laptop:/home/roopa/Desktop/project/boa-0.94.13# ls /etc/boa/boa.conf
/etc/boa/boa.conf
root@roopa-laptop:/home/roopa/Desktop/project/boa-0.94.13# boa -c /etc &
[1] 11675
Could not open boa.conf for reading.
root@roopa-laptop:/home/roopa/Desktop/project/boa-0.94.13# boa -c /etc/boa/
[06/Sep/2012:17:14:23 +0000] boa.c:226 - icky Linux kernel bug!: Success
[1]+  Exit 1                  boa -c /etc
root@roopa-laptop:/home/roopa/Desktop/project/boa-0.94.13# ls
boa.conf   contrib  docs     extras         README  test.cgi
ChangeLog  CREDITS  examples  Gnu_License  src
root@roopa-laptop:/home/roopa/Desktop/project/boa-0.94.13# cd src/
root@roopa-laptop:/home/roopa/Desktop/project/boa-0.94.13/src# ls
acconfig.h      cgi_header.o          escape.h      Makefile       select.c
aclocal.m4      cgi.o                 escape.o      Makefile.in    select.o
alias.c         check_struct_for.m4   get.c         mmap_cache.c   signals.c
alias.o         compat.h              get.o         mmap_cache.o   signals.o
boa             config.c              globals.h     parse.h        sublog.c
boa.c           config.cache          hash.c        pipe.c         sublog.o
boa_grammar.y   config.h              hash.o        pipe.o         test.cgi
boa.h           config.h.in           index_dir.c   queue.c        timestamp.c
boa_indexer     config.log            index_dir.o   queue.o        timestamp.o
boa_lexer.l     config.o              ip.c          read.c         util.c
boa.o           config.status         ip.o          read.o         util.o
buffer.c        configure             lex.yy.c      request.c      webindex.pl
buffer.o        configure.in          lex.yy.o      request.o      y.tab.c
cgi.c           defines.h             log.c         response.c     y.tab.h
cgi_header.c    escape.c              log.o         response.o     y.tab.o
root@roopa-laptop:/home/roopa/Desktop/project/boa-0.94.13/src# cd /
root@roopa-laptop:/# ls
bin       dev          initrd.img.old  mime.types  root     sys  vmlinuz
boa.conf  etc          lib             mnt         sbin     tmp  vmlinuz.old
boot      home         lost+found      opt         selinux  usr  web
cdrom     initrd.img   media           proc        srv      var
root@roopa-laptop:/# ls /web/
cgi-bin
```

Fig. 4.1: Boa configuration



```
Partially written block 750 detected
Partially written block 750 detected
Partially written block 750 detected
Partially written block 750 detected
Partially written block 750 detected
Partially written block 750 detected
Partially written block 750 detected
Partially written block 750 detected
Partially written block 750 detected
Partially written block 750 detected
Partially written block 750 detected
Partially written block 750 detected
Partially written block 750 detected
Partially written block 750 detected
Partially written block 705 detected
Partially written block 705 detected
Partially written block 705 detected
Partially written block 705 detected
Partially written block 705 detected
Partially written block 705 detected
Partially written block 705 detected
Partially written block 705 detected
Partially written block 705 detected
Partially written block 705 detected
Partially written block 705 detected
yaffs_read_super: isCheckpointed 0
VFS: Mounted root (yaffs filesystem) on device 31:3.
Freeing init memory: 156K
hwclock: settimeofday() failed: Invalid argument
[26/Apr/1974:20:38:19 +0000] boa: server version Boa/0.94.13
[26/Apr/1974:20:38:19 +0000] boa:
[26/Apr/1974:20:38:19 +0000] boa: starting server pid=681, port 80

Try to bring eth0 interface up......eth0: link down
Done

Please press Enter to activate this console.
[root@FriendlyARM /]# eth0: link up, 100Mbps, full-duplex, lpa 0x45E1

[root@FriendlyARM /]#
```

Fig. 4.2: Boa ported to the development board

V. RESULTS AND DISCUSSIONS

An application program is written in C, which uses the basic SQLite functions for accepting the data, storing and modifying it in the database and display it in a particular format. The flow chart of the application program is as shown in fig 5.1. When the application is run with the data given as shown in Fig 5.2, the output is as shown in Fig 5.3.

Boa web server implementation: In order to implement the web server, it is required to know the server (target board) ip address. The ip address of target board can be known by the command

```
# ifconfig
```

All the required sqlite commands for creating a database are placed in a sql script and named as test1.sql. The commands for updating the database and displaying the database are placed in a shell script, named as cmnd.sh. The commands for copying the database file to a .html file in /www directory are placed in a shell script named as exe.sh. The commands are as shown in Fig 5.6. When the client requests for a file, the server can send it only when it contains the file in /www directory. The client has to request for the page that is present in /www directory of server by typing the server address and specifying the file name or path.

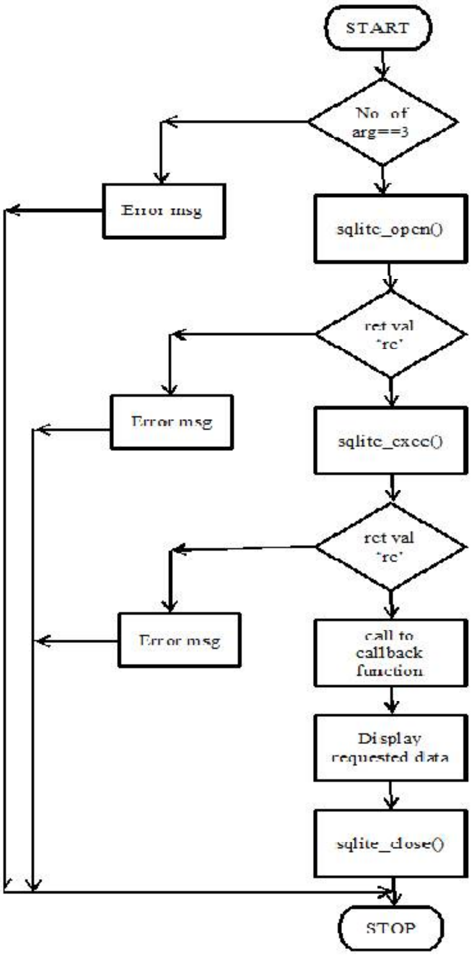


Fig. 5.1: Application program

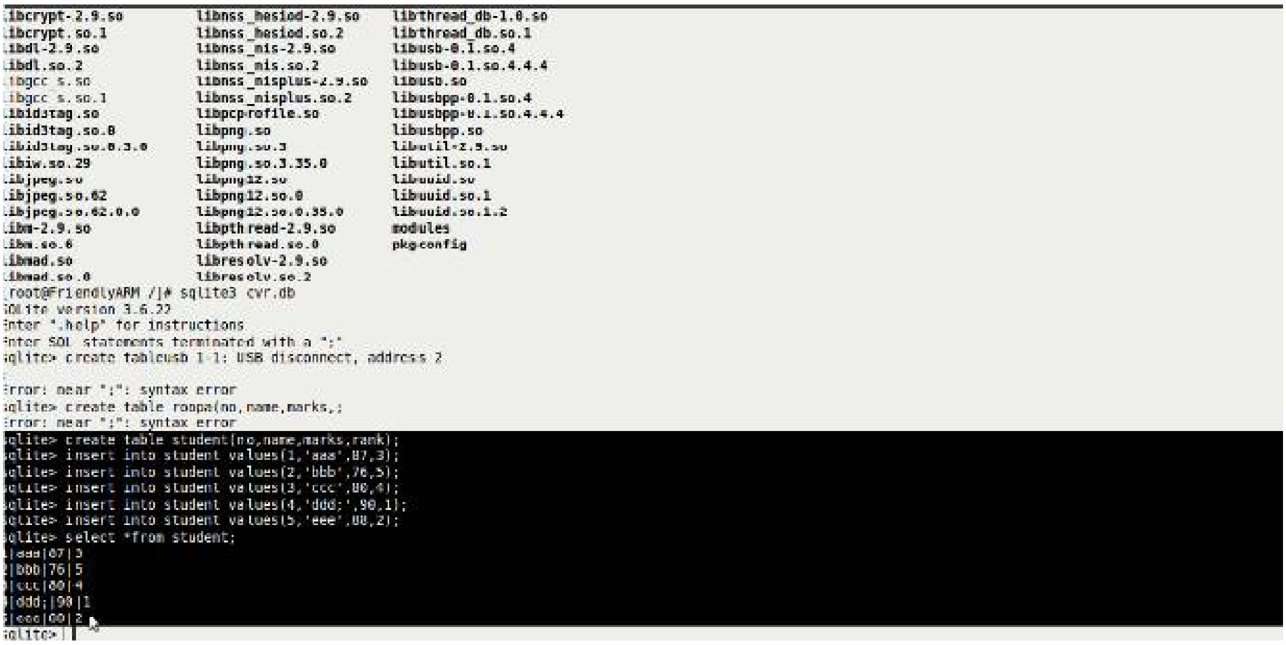
DOI:10.15662/IJAREEIE.2014.0302057

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

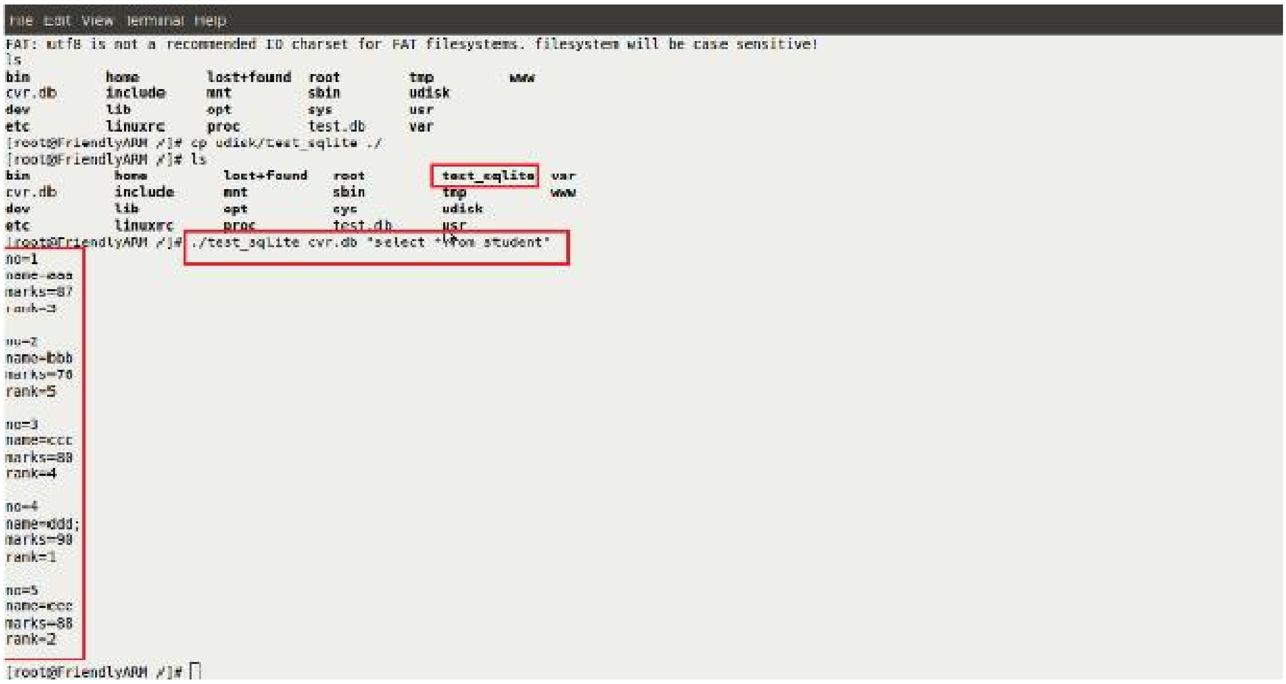
(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2014



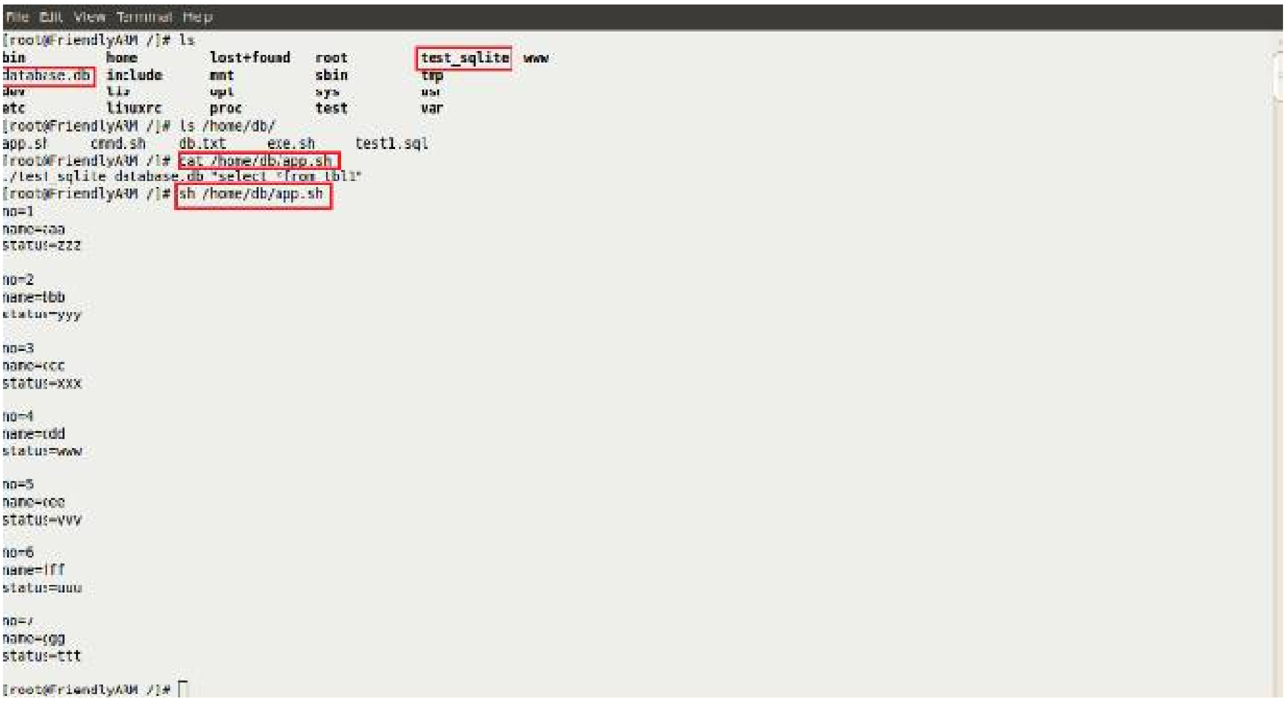
```
.ibcrypt-2.9.so          libnss_hesiod-2.9.so    libthread_db-1.0.so
.ibcrypt.so.1            libnss_hesiod.so.2      libthread_db.so.1
.ibdl-2.9.so             libnss_nis-2.9.so       libusb-0.1.so.4
.ibdl.so.2               libnss_nis.so.2         libusb-0.1.so.4.4.4
.ibgcc_s.so              libnss_nisplus-2.9.so   libusb.so
.ibgcc_s.so.1            libnss_nisplus.so.2     libusbpp-0.1.so.4
.ibid3tag.so             libpcprofile.so         libusbpp-0.1.so.4.4.4
.ibid3tag.so.0           libpng.so               libusbpp.so
.ibid3tag.so.0.3.0       libpng.so.3             libutil-2.9.so
.ibiw.so.29              libpng.so.3.35.0        libutil.so.1
.ibjpeg.so               libpng12.so             libuuid.so
.ibjpeg.so.62            libpng12.so.0           libuuid.so.1
.ibjpeg.so.62.0.0        libpng12.so.0.35.0      libuuid.so.1.2
.ibm-2.9.so              libpthread-2.9.so       modules
.ibm.so.6                libpthread.so.0         pkgconfig
.ibmad.so                libresolv-2.9.so
.ibmad.so.0              libresolv.so.2
.root@FriendlyARM /]# sqlite3 cvr.db
SQLite version 3.6.22
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> create tableusb 1-1: USB disconnect, address 2
;
Error: near ";": syntax error
sqlite> create table roopa(no,name,marks,;
Error: near ";": syntax error
sqlite> create table student(no,name,marks,rank);
sqlite> insert into student values(1,'aaa',87,3);
sqlite> insert into student values(2,'bbb',76,5);
sqlite> insert into student values(3,'ccc',80,4);
sqlite> insert into student values(4,'ddd;',90,1);
sqlite> insert into student values(5,'eee',88,2);
sqlite> select *from student;
1|aaa|87|3
2|bbb|76|5
3|ccc|80|4
4|ddd;|90|1
5|eee|88|2
sqlite> |
```

Fig.5.2: SQLite implementation



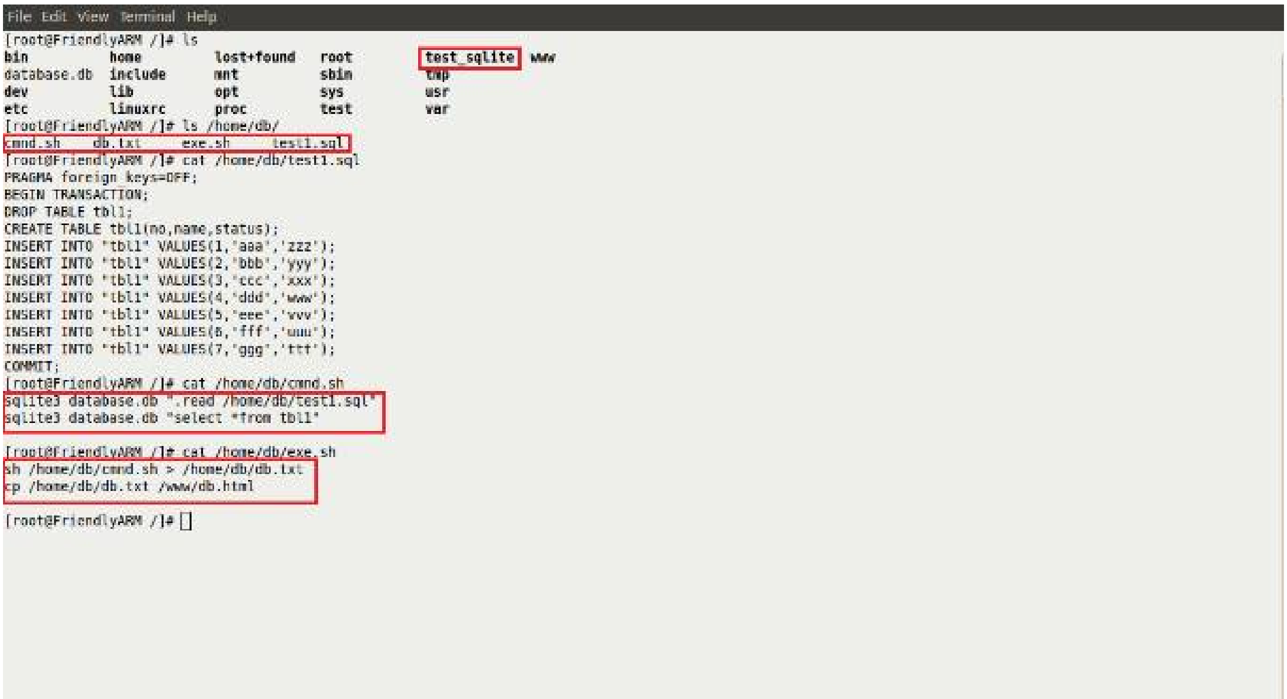
```
FAT: utf8 is not a recommended IO charset for FAT filesystems. filesystem will be case sensitive!
ls
bin         home        lost+found  root        tmp         www
cvr.db      include     mnt         sbin        udisk
dev         lib         opt         sys         usr
etc         linuxrc     proc        test.db     var
[root@FriendlyARM /]# cp udisk/test_sqlite ./
[root@FriendlyARM /]# ls
bin          home         lost+found   root         test_sqlite  var
cvr.db       include      mnt          sbin         tmp          www
dev          lib          opt          sys          udisk
etc          linuxrc      proc         test.db      usr
[root@FriendlyARM /]# ./test_sqlite cvr.db "select *from student"
no=1
name=aaa
marks=87
rank=3

no=2
name=bbb
marks=76
rank=5

no=3
name=ccc
marks=80
rank=4

no=4
name=ddd;
marks=90
rank=1

no=5
name=eee
marks=88
rank=2

[root@FriendlyARM /]# 
```

Fig. 5.3: Output of the application program



```
[root@FriendlyARM /]# ls
bin          home        lost+found  root        test_sqlite  www
database.db  include     mnt         sbin        tmp
dev          lib         opt         sys         usr
etc          linuxrc     proc        test        var
[root@FriendlyARM /]# ls /home/db/
app.sh     cmnd.sh    db.txt     exe.sh     test1.sql
[root@FriendlyARM /]# cat /home/db/app.sh
./test_sqlite database.db "select *from tbl1"
[root@FriendlyARM /]# sh /home/db/app.sh
no=1
name=aaa
status=zzz

no=2
name=bbb
status=yyy

no=3
name=ccc
status=xxx

no=4
name=ddd
status=www

no=5
name=eee
status=vvv

no=6
name=fff
status=uuu

no=7
name=ggg
status=ttt

[root@FriendlyARM /]# 
```

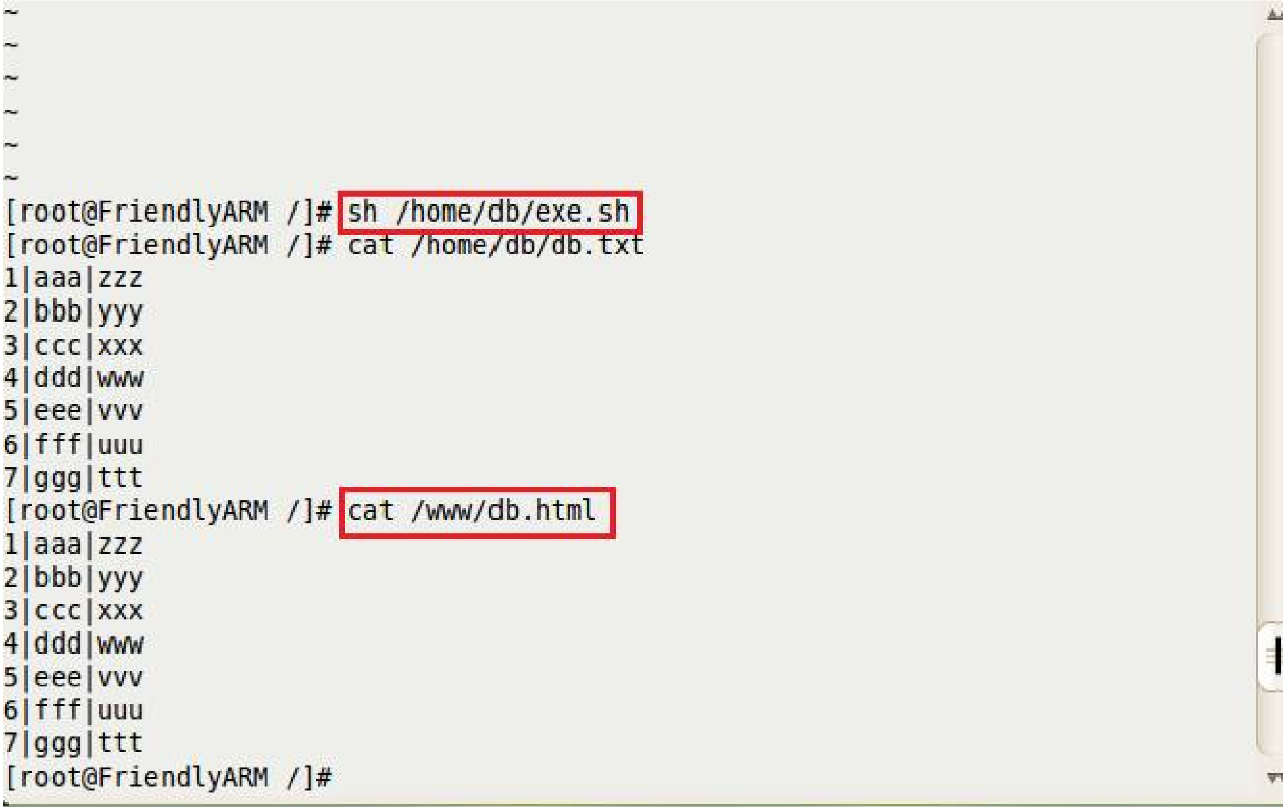
Fig. 5.4: Executing the application program using shell script



```
[root@FriendlyARM /]# ls
bin          home        lost+found  root        test_sqlite  www
database.db  include     mnt         sbin        tmp
dev          lib         opt         sys         usr
etc          linuxrc     proc        test        var
[root@FriendlyARM /]# ls /home/db/
cmnd.sh    db.txt     exe.sh     test1.sql
[root@FriendlyARM /]# cat /home/db/test1.sql
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
DROP TABLE tbl1;
CREATE TABLE tbl1(no,name,status);
INSERT INTO "tbl1" VALUES(1,'aaa','zzz');
INSERT INTO "tbl1" VALUES(2,'bbb','yyy');
INSERT INTO "tbl1" VALUES(3,'ccc','xxx');
INSERT INTO "tbl1" VALUES(4,'ddd','www');
INSERT INTO "tbl1" VALUES(5,'eee','vvv');
INSERT INTO "tbl1" VALUES(6,'fff','uuu');
INSERT INTO "tbl1" VALUES(7,'ggg','ttt');
COMMIT;
[root@FriendlyARM /]# cat /home/db/cmnd.sh
sqlite3 database.db ".read /home/db/test1.sql"
sqlite3 database.db "select *from tbl1"

[root@FriendlyARM /]# cat /home/db/exe.sh
sh /home/db/cmnd.sh > /home/db/db.txt
cp /home/db/db.txt /www/db.html

[root@FriendlyARM /]# 
```

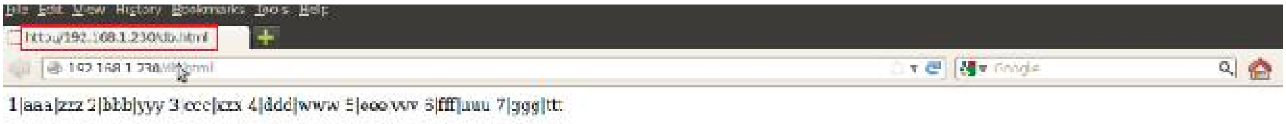
Fig. 5.6: The shell and sql scripts used in webserver implementation

DOI:10.15662/IJAREEIE.2014.0302057



```
~
~
~
~
~
~
[root@FriendlyARM /]# sh /home/db/exe.sh
[root@FriendlyARM /]# cat /home/db/db.txt
1|aaa|zzz
2|bbb|yyy
3|ccc|xxx
4|ddd|www
5|eee|vvv
6|fff|uuu
7|ggg|ttt
[root@FriendlyARM /]# cat /www/db.html
1|aaa|zzz
2|bbb|yyy
3|ccc|xxx
4|ddd|www
5|eee|vvv
6|fff|uuu
7|ggg|ttt
[root@FriendlyARM /]#
```

Fig. 5.7: Updating the database by executing exe.sh


File Edit View History Bookmarks Tools Help
http://192.168.1.230/db.html
192.168.1.230/db.html
Google

```
1|aaa|zzz 2|bbb|yyy 3|ccc|xxx 4|ddd|www 5|eee|vvv 6|fff|uuu 7|ggg|ttt
```

***Fig. 5.8: html page containing** the database, sent to the client on request*

VI. CONCLUSION AND FUTURE SCOPE

The SQLite database is designed and implemented for embedded platform based on the ARM-Linux operating system. The web server boa has been ported to arm linux platform and is implemented. When compared to other databases, SQLite is perfectly suitable for embedded applications as it has the advantages like zero configurations, server less, variable length record, cross platform, manifest typing, compact size etc. Similarly, when compared to the traditional PC server, boa server has small storage, low cost, portability, easy to maintain and upgrade. The web server Boa is selected for the proposal, because it consumes low power that is suitable for embedded applications. It also has more functionsand supports CGI communication between external expansion applications and web server, which can be achieved through CGI technology.

The SQLite database along with embedded web server can be applied easily to embedded fields such as on-site AC servo system, industrial control, and intelligent appliances. Remote monitoring is applicable in a wide range of industries like the oil and gas industry pharmaceutical, rail networks, electricity transmission and distribution of food

DOI:10.15662/IJAREEIE.2014.0302057

and beverages.For example, we consider the upstream part of the oil and gas industry. It usually has several remote locations which have a lot of potential for optimizing the manual activities done at the sites. Remote monitoring and maintaining the database is a solution to achieve this. The benefits of using remote monitoring and maintaining the database are adherence to regulatory, requirements in Operations, Improved safety in gas pipelines and plant area, handling hydrocarbon (explosive fluids), cost benefits in centralized remote operations, availability of real time data for better decisions, minimize the risk of emergency shutdowns due to failures and extend or eliminate scheduled service intervals.

REFERENCES

[1] Guang Dong, Wei He, Yuhang Wang, Database Design on Embedded Home Gateway and Web Server Implementation
[2] Neil Matthew, Richard Stones, Beginning Linux@Programming, Wiley Publishing, Inc., 3rd edition.
[3] Christopher Hallinan, Embedded Linux Primer, APractical Real-World Approach, Prentice Hall, 2nd edition.
[4] Michael Owens, Embedding an SQL Database with SQLite, Linux Journal, June2003
[5] SUN Yu-hong, CUI Shao-bin, Research on embedded server of home network, Computer Engineering and Design, vol.29Antenna", IEEE transactions on antennas and propagation, vol.61, no. 3, march 2013
[6] Chen Tian-huang, Wuhan, Huang Jia, Design and Realization of CGI in Embedded Dynamic Web Technology