Enhanced Embedded Linux Board Support Package Field Upgrade – A Cost Effective Approach

Kantam Nagesh[1], Oeyvind Landsnes[2], Tore Fuglestad[2], Nina Svensen[2] and Deepak Singhal[1]

[1]ABB Ability Innovation Center, Robotics and Motion, Bengaluru, India
[2]ABB AS, Robotics and Motion, Bryne, Norway

ABSTRACT

Latest technology, new features and kernel bug fixes shows a need to explore a cost-effective and quick upgradation of Embedded Linux BSP of Embedded Controllers to replace the existing U-Boot, Linux kernel, Dtb file, and JFFS2 File system. This field upgrade technique is designed to perform an in-the-field flash upgrade while the Linux is running. On successful build, the current version and platform specific information will be updated to the script file and further with this technique the file system automates the upgrade procedure after validating for the version information from the OS-release and if the version is different it will self-extract and gets installed into the respective partitions. This Embedded Linux BSP field upgrade invention is more secured and will essentially enable the developers and researchers working in this field to utilize this method which can prove to be cost-effective on the field and beneficial to the stake holder.

KEYWORDS

Embedded Linux, Upgrade, Kernel, U-boot, JFFS2 File system.

INTRODUCTION

Embedded systems form an essential part of the connected world. 1, 2 BSP (Board Support Package) is a collection of binary code and supported files which are used to create a Linux kernel firmware and filesystem images for a particular target. Latest advancements in field upgrade is of prime importance to enhance the BSP upgradation. Embedded Systems are essentially designed for a specific task based on its characterization. 1 Building embedded systems, Linux based or otherwise, involves a lot of effort. The conventional method of updating the configuration or performing software upgrade of Linux system is a nonsequitur in the embedded space. Many developers use these cumbersome methods for lack of a better alternative. There is a need to improve the design and important aspects of the system as its performance, real time constraints, hardware interfaces and cost. 3

Today the conventional method for Embedded Linux BSP field upgrade is being used globally in various products which is time consuming and cumbersome. This paper proposes a cost effective and quicker BSP field upgrade technique for Embedded Linux OS which reduces the technician time from performing 12 steps used in the conventional method to only 2 steps reducing the time taken for the upgrade, cost and also the manpower.

Methodology

1. UPDATE TOOLS REQUIRED

- Makeself
- Teraterm
- FTP Server
- FTP Client
- Secured proprietary application / Robot Studio

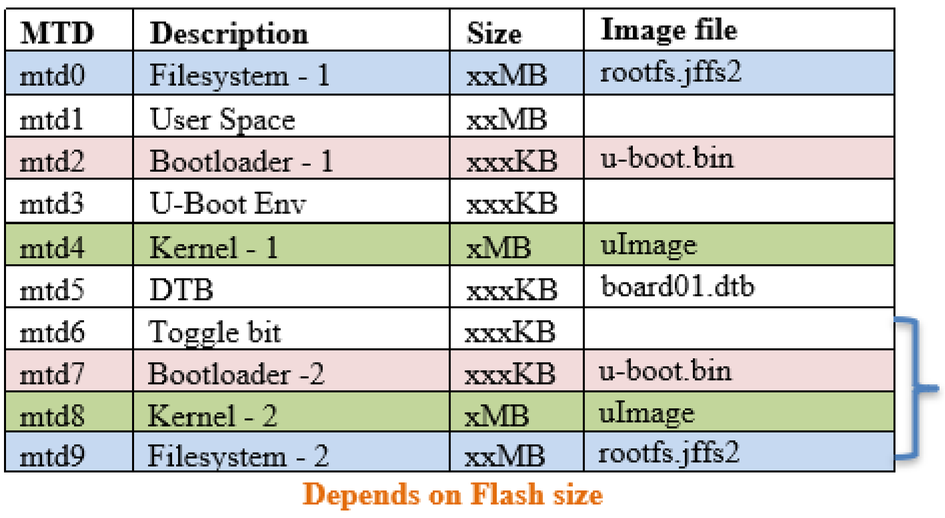
2. SUPPORT FILES

- u-boot.bin
- uImage
- rootfs.jffs2
- board01.dtb
- 01_uboot.install
- 02_linux.install

3. ADDRESS MAP

Flash to program the Embedded Linux BSP is partitioned as given in Table 1

Table 1: Partitions



MTD	Description	Size	Image file
mtd0	Filesystem - 1	xxMB	rootfs.jffs2
mtd1	User Space	xxMB	
mtd2	Bootloader - 1	xxxKB	u-boot.bin
mtd3	U-Boot Env	xxxKB	
mtd4	Kernel - 1	xMB	uImage
mtd5	DTB	xxxKB	board01.dtb
mtd6	Toggle bit	xxxKB	
mtd7	Bootloader -2	xxxKB	u-boot.bin
mtd8	Kernel - 2	xMB	uImage
mtd9	Filesystem - 2	xxMB	rootfs.jffs2

Depends on Flash size

4. SETTING UP

a. To begin with setting up the system for field upgradation and to see the log messages while upgrading

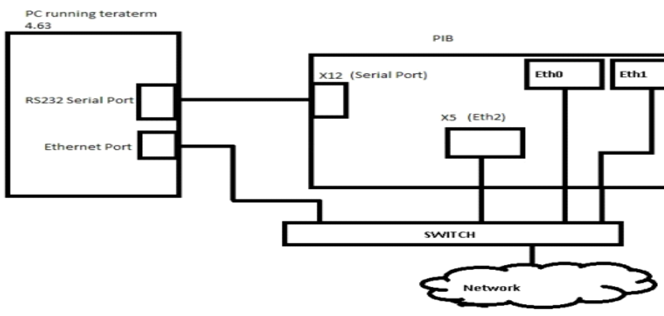
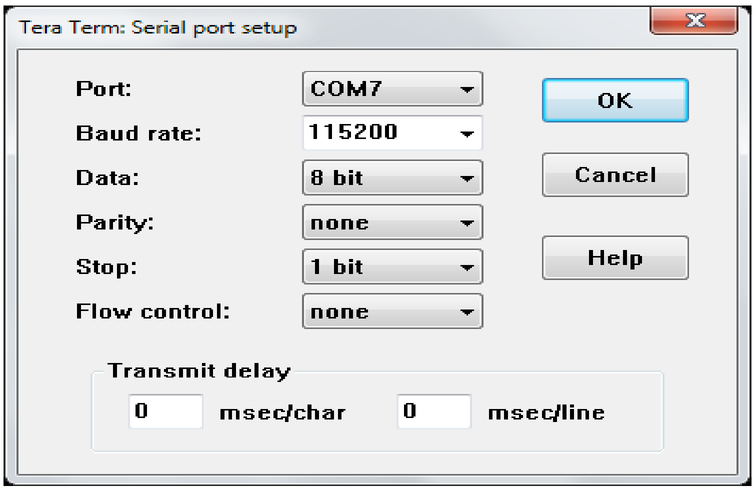


Fig 1:_Serial and Ethernet Connection diagram between Embedded Linux Controller and Laptop/Desktop

B. Serial Configuration

Fig 2 shows the terminal which should be set up for RS232 communication at 115200 baudrate, 8data bits, no parity, one stop bit, and XON/XOFF flow control.



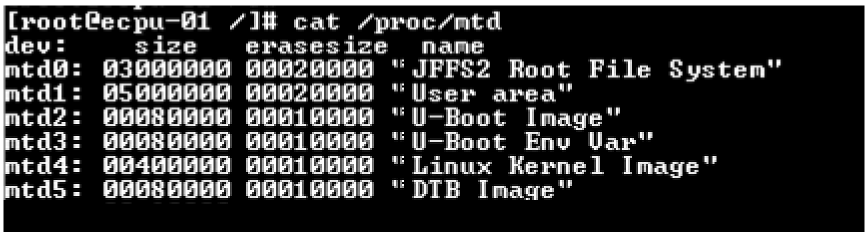
Tera Term: Serial port setup

Port: COM7
Baud rate: 115200
Data: 8 bit
Parity: none
Stop: 1 bit
Flow control: none

Transmit delay
0 msec/char 0 msec/line

Fig 2: Serial configuration

C. Flash Organization



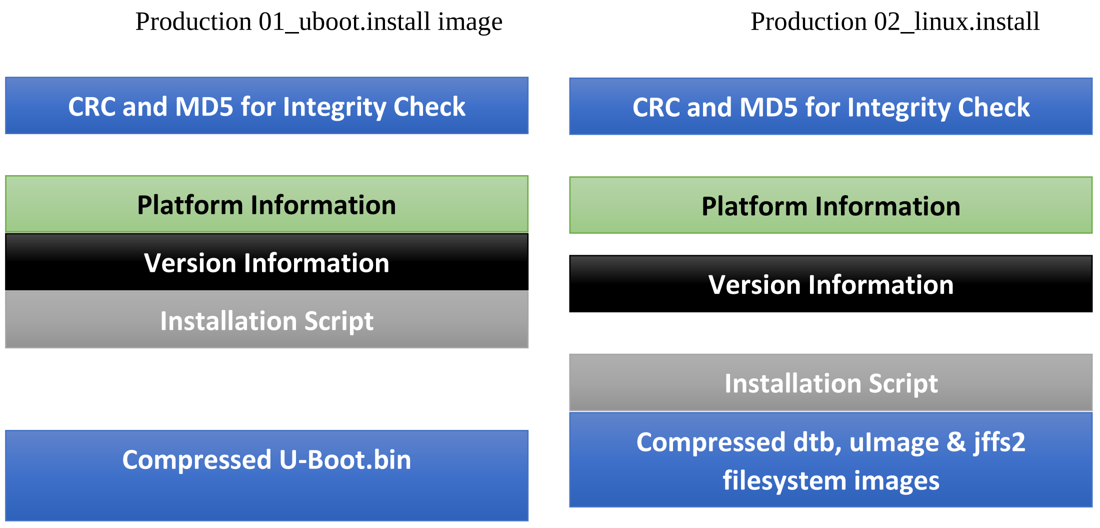
```
[root@ecpu-01 /]# cat /proc/mtd
dev:    size   erasesize  name
mtd0: 03000000 00020000 "JFFS2 Root File System"
mtd1: 05000000 00020000 "User area"
mtd2: 00080000 00010000 "U-Boot Image"
mtd3: 00080000 00010000 "U-Boot Env Var"
mtd4: 00400000 00010000 "Linux Kernel Image"
mtd5: 00080000 00010000 "DTB Image"
```

Fig 3: Flash Partition table

5. Procedure to Build the Linux BSP Binaries Using Buildroot for Embedded Board

- git clone git@xxx.xxx.xxx.xxx:root/xyz.git (from local develop branch)
- cd buildroot
- git checkout development branch
- git pull
- make clean
- make defconfig
- make

Buildroot will generate u-boot.bin, u-boot.bin, uImage, rootfs.jffs2, board01.dtb, **01_uboot.install** and **02_linux.install** files in “output/images” folder


Production 01_uboot.install image
CRC and MD5 for Integrity Check
Platform Information
Version Information
Installation Script
Compressed U-Boot.bin
Production 02_linux.install
CRC and MD5 for Integrity Check
Platform Information
Version Information
Installation Script
Compressed dtb, uImage & jffs2 filesystem images

6. Conventional Upgrade Patterns

Previous Upgrade Procedure for Embedded Linux Boards with Network

a. Using commands in u-boot (Manual)

1) Connect the serial port of the board to serial port of PC
2) Start the tftp server and point the TFTP directory to: <Release_Name>\release Where <Release_Name> is the release to which the BSP is being upgraded.
3) Start the teraterm application
4) Power on the board
5) Wait for the message “*Hit ^C (Control C) to stop autoboot*” and press Ctrl+C till you get the IUP>
6) On getting the IUP> enter the following commands:
7) To set envirnonment variables:
   IUP> setenv ethaddr <HW address of the board>
   IUP> setenv ipaddr <Required IP address>
   IUP> setenv serverip <IP address of PC running TFTP server>
8) To copy rootfs
   IUP> tftp xxxxxxxx rootfs.jffs2
   IUP> nand erase 0 yyyyyyyy

```
IUP> nand write.jffs2 xxxxxxxx 0 $filesize
```

9) To copy kernel image

```
IUP> tftp xxxxxxxx uImage
IUP> sf erase yyyyyyyy zzzzzzzz
IUP> sf write xxxxxxxx yyyyyyyy $filesize
```

10) To copy dtb

```
IUP> tftp xxxxxxxx board.dtb
IUP> sf erase yyyyyyyy zzzzzzzz
IUP> sf write xxxxxxxx yyyyyyyy $filesize
```

11) To copy u-boot

```
IUP> sf erase yyyyyyyy zzzzzzzz
IUP> tftp xxxxxxxx u-boot-spi.bin
IUP> sf write xxxxxxxx yyyyyyyy $filesize
```

12) Reset the board

```
IUP> reset
```

13) After this reset the board will boot through to kernel prompt.
14) On bootup, in the kernel prompt, enter the command:

```
uname –a
```

The output of the command will contain the Linux version followed by the release name. eg:

```
Linux node1 3.18.9-rt4 #1 PREEMPT RT Fri Nov 24 09:44:31 IST 2017 ppc GNU/Linux
```

This name should match the release name of the BSP to which the TFTP server points.

b. Using TTL Scripts in u-boot (Partially automated)

Same way binaries can be upgraded from u-boot prompt and it can be reducible by few steps but not fully automated. We need have flash related utilities (nand, sf and cp.b) enabled in u-boot and manual interference is required.

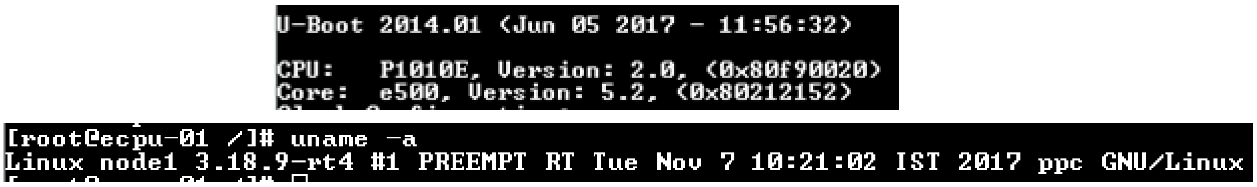
Many developers use these cumbersome methods for lack of a better alternative as they get harder as the number of devices increase. As this was very time consuming and is a method, which is either fully manual or partially automated, technique making the product/device work slower and using more of manpower. Hence there was a need for thorough research on this with many techniques tested that led to the invention of the below mentioned new technique for field upgradation of the Embedded Linux BSP (OS system), this reduced the conventional method with 12 or more steps to only 2 steps proving it to be a very cost-effective, more importantly secured and easy approach which can be beneficial for the developers, stake holders and customers at large. The process is as shown below;

7. Cost-effective and quicker Automated upgradation technique

This idea reduces the technician time from 12 or more steps used in the conventional method to only 2 steps in this new approach of field upgrade method.

The Process is as follows;

The Robot Controller board is booted and running with Linux x.xx.xx version, from linux shell it is fairly easy to upgrade u-boot, dtb and kernel uImage with the help of linux mtd utilities but the same is not easy for upgrading the filesystem. To initiate the Linux BSP Upgrade process, generate 01_uboot.install and 02_linux.install files. On successful build, the current version and target platform information will be updated to the script file (which is included in the .install file).



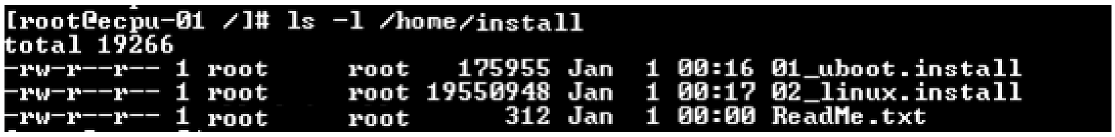
```
U-Boot 2014.01 (Jun 05 2017 - 11:56:32)

CPU:   P1010E, Version: 2.0, (0x80f90020)
Core:  e500, Version: 5.2, (0x80212152)
```

```
[root@ecpu-01 /]# uname -a
Linux node1 3.18.9-rt4 #1 PREEMPT RT Tue Nov 7 10:21:02 IST 2017 ppc GNU/Linux
```

Fig 4: Before upgrade, screenshots of the U-Boot and Linux kernel version

Step1. Copy 01_uboot.install & 02_linux.install files into /home/install/ folder into the Linux Controller filesystem using tftp/scp/sftp/Secured proprietary application.
e.g., tftp -g -r 02_linux.install <tftp server ip>

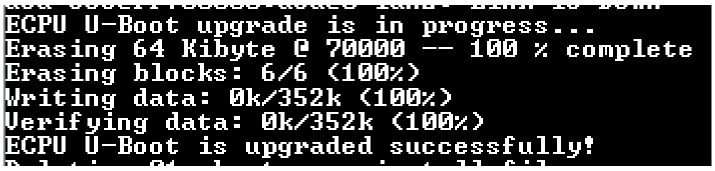


```
[root@ecpu-01 /]# ls -l /home/install
total 19266
-rw-r--r-- 1 root     root   175955 Jan  1 00:16 01_uboot.install
-rw-r--r-- 1 root     root 19550948 Jan  1 00:17 02_linux.install
-rw-r--r-- 1 root     root      312 Jan  1 00:00 ReadMe.txt
```

Fig 5: After copying files in the filesystem

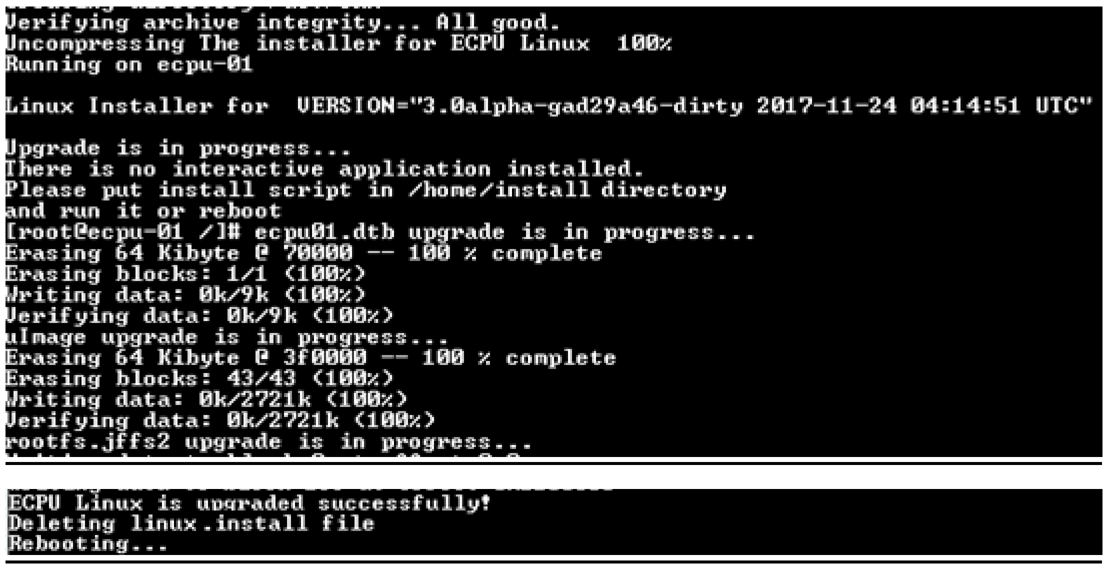
Step2. On Reboot xxxinstall startup script will look for *.install files and it will validate with the current platform and version information from os-release file for linux and the version information from u-boot.bin for uboot and if the version is different it will **self-extract** and **copies all the required utilities** into the **shared memory** (dev/shm) and it will stop all running processes/services and gets installed into the respective partitions (u-boot, Linux kernel, DTB and JFFS2 Filesystem) using mtd utilities. Similarly when version is same or platform is different then installer will show a message and exit.

Once installation is complete it will delete *.install files and then system reboots automatically with the newer version.



```
ECPU U-Boot upgrade is in progress...
Erasing 64 Kibyte @ 70000 -- 100 % complete
Erasing blocks: 6/6 (100%)
Writing data: 0k/352k (100%)
Verifying data: 0k/352k (100%)
ECPU U-Boot is upgraded successfully!
```

Fig 6: Below screenshot is while upgrading U-Boot



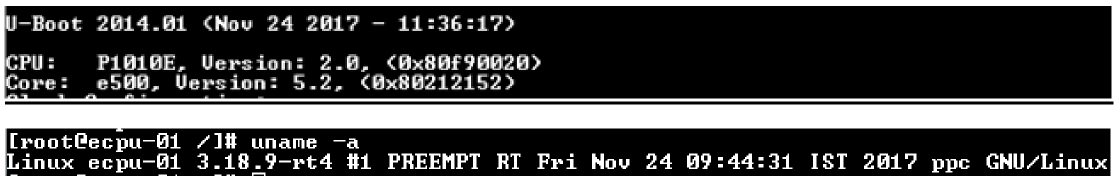
```
Verifying archive integrity... All good.
Uncompressing The installer for ECPU Linux  100%
Running on ecpu-01

Linux Installer for  VERSION="3.0alpha-gad29a46-dirty 2017-11-24 04:14:51 UTC"

Upgrade is in progress...
There is no interactive application installed.
Please put install script in /home/install directory
and run it or reboot
[root@ecpu-01 /]# ecpu01.dtb upgrade is in progress...
Erasing 64 Kibyte @ 70000 -- 100 % complete
Erasing blocks: 1/1 (100%)
Writing data: 0k/9k (100%)
Verifying data: 0k/9k (100%)
uImage upgrade is in progress...
Erasing 64 Kibyte @ 3f0000 -- 100 % complete
Erasing blocks: 43/43 (100%)
Writing data: 0k/2721k (100%)
Verifying data: 0k/2721k (100%)
rootfs.jffs2 upgrade is in progress...
```

```
ECPU Linux is upgraded successfully!
Deleting linux.install file
Rebooting...
```

Fig 7: Below screenshot is while upgrading device tree, kernel and filesystem



```
U-Boot 2014.01 (Nov 24 2017 - 11:36:17)

CPU:   P1010E, Version: 2.0, (0x80f90020)
Core:  e500, Version: 5.2, (0x80212152)
```

```
[root@ecpu-01 /]# uname -a
Linux ecpu-01 3.18.9-rt4 #1 PREEMPT RT Fri Nov 24 09:44:31 IST 2017 ppc GNU/Linux
```

Fig 8: After upgrade, screenshots of the U-Boot and Linux kernel version

8. SECURE FLASH PROCESS FLOW DIAGRAM

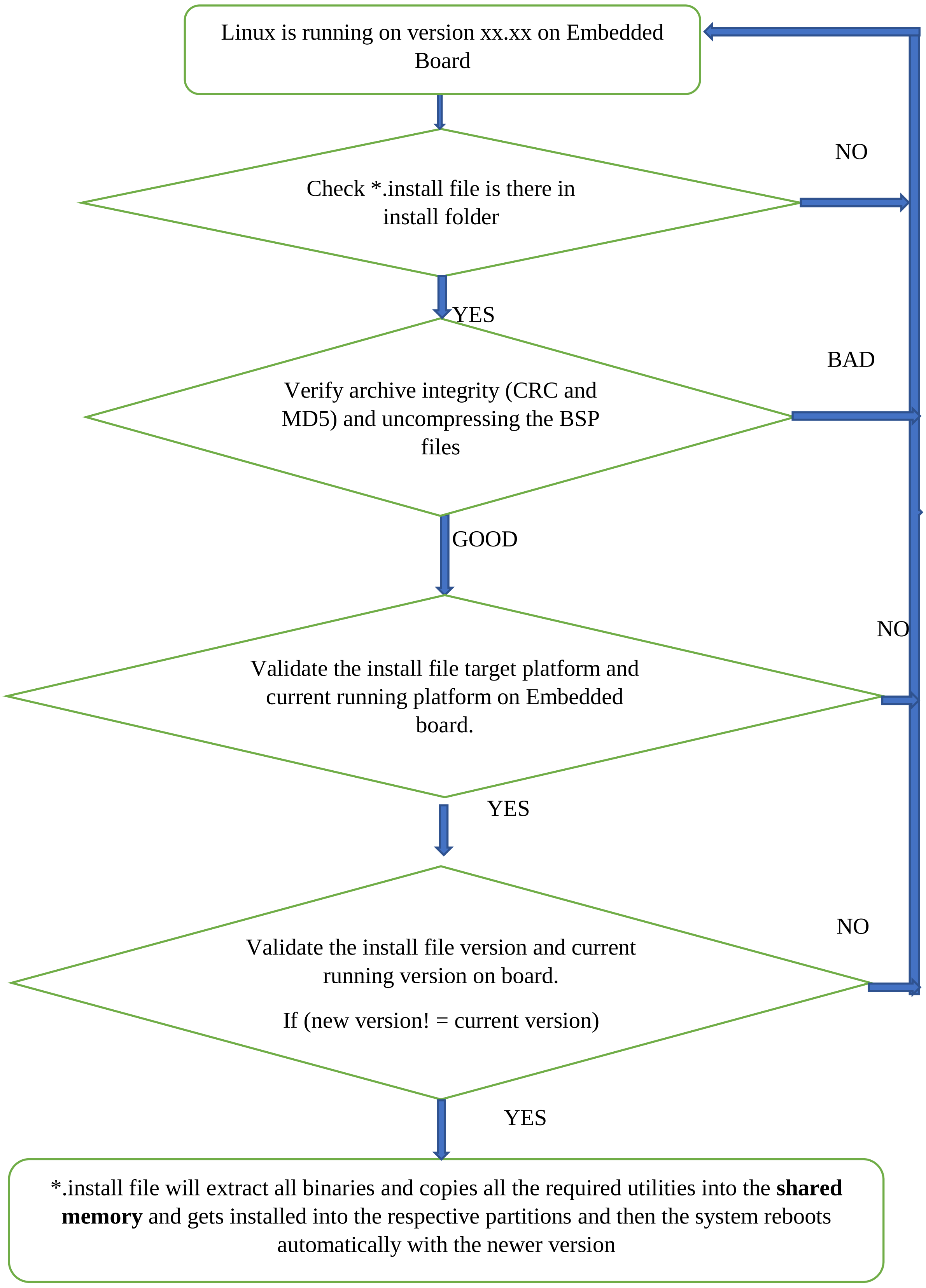

Linux is running on version xx.xx on Embedded Board
Check *.install file is there in install folder
NO
YES
Verify archive integrity (CRC and MD5) and uncompressing the BSP files
BAD
GOOD
Validate the install file target platform and current running platform on Embedded board.
NO
YES
Validate the install file version and current running version on board.
If (new version! = current version)
NO
YES
*.install file will extract all binaries and copies all the required utilities into the shared memory and gets installed into the respective partitions and then the system reboots automatically with the newer version

Fig 9: Secure Flash Process Flow Diagram

9. NOVEL FEATURES OF THIS TECHNIQUE DIFFERENT FROM THE CONVENTIONAL; AUTOMATED EMBEDDED LINUX BSP UPGRADATION:

1. It allows upgrade of the following components -
   - uImage: This is the image of realtime linux kernel.
   - u-boot.bin: This is the bootloader binary.
   - rootfs.jffs2: This is Linux Root File system image which contains the entire directory structue alongwith all applications.
   - xxxx-01.dtb: The device tree file corresponding to the board.
2. Field upgrade does not need the user to get into the bootloader (u-boot) prompt. It happens on the kernel prompt through a bash script

3. It automatically reboots the system after the upgrade to boot up with the new binaries.

4. In case of any errors (flash write errors or corrupted binaries), the system can be booted up with the previous binaries by toggling the bit in /dev/mtdblockX incase sufficient flash(s)/partitions are available,

Thus this technique proves to be better, more secured and quicker than the conventional method and can be cost effective on field and beneficial to the customer/stakeholder to enhance the performance of the Embedded Linux operating controllers.

Technical Feasibility

The invention has been tested on the Embedded Linux operating Robot controllers and here is the results when it was compared to the conventional upgradation.

Convention upgradation: It took 30-40 minutes (manual interference) for one Embedded Linux operating controllers to upgrade while Linux is running.

Invention: It takes 3-4 minutes for one Embedded Linux operating controllers to upgrade as it is fully automated and there is no manual interference and it is secured and is almost 10times faster and cost-effective using lesser man-hours and manpower. Such technique might be a control unit firmware update for fixing bugs, improving features.

Security Features

This technique is made secure by accepting only original softwares for embedded system updates, alteration of data permitted only to authenticated parties and in case of errors while updating, the validation and verification to be done through digital signatures, certification and cryptography to maintain the integrity and authenticity.

CONCLUSION

This invention has additional features like validation of the install file target platform and current running platform on embedded board, verification by conducting CRC and MD5 integrity checks and installation of all Linux BSP binaries in a completely secure way. This proves to be better, more secured and quicker than the conventional method, cost effective on the field as it has minimal manual interference, shorter time and lesser number of steps needed to perform the upgrade, beneficial to the customer/stakeholder to enhance the performance of the Embedded Linux operating controllers. In the future use of other encryption methods, digital signatures and authentication can be recommended to achieve better security in several software modules.

REFERENCES

1 https://www.allegrosoft.com/wp-content/uploads/Secure-Firmware-Updates-Paper.pdf

2 Kumar Eswar, M. Kamaraju, Yadav Kumar Ashok. Porting and BSP Customization of Linux on ARM Platform. International Journal of Computer Applications, October 2013; 80 (15): 36-42.

3 https://blog.feabhas.com/wp-content/uploads/2014/11/Writing-a-Linux-Update-Mechanism.pdf

4 https://www.xilinx.com/support/documentation/application_notes/xapp1146.pdf

5 https://patents.google.com/scholar/135927621267806534?q=linux&q=BSP&q=upgrade&scholar

6 https://patents.google.com/patent/US5794052A/en?q=linux&q=BSP&q=upgrade&scholar

7 http://www.linuxjournal.com/content/updating-firmware-linux-based-devices

8 http://events.linuxfoundation.org/sites/events/files/slides/linuxcon-japan-2016-softwre-updates-sangorrin.pdf

9 https://lists.linuxfoundation.org/pipermail/automotive-discussions/2016-May/002061.html

10 https://wiki.openwrt.org/doc/howto/generic.sysupgrade

11 http://www.ti.com/lit/an/slaa682/slaa682.pdf

12 https://www.embedded.com/design/configurable-systems/4008264/Using-software-flashing-to-secure-embedded-device-updates

13 https://ieeexplore.ieee.org/document/7807959

14 https://www.st.com/content/ccc/resource/sales_and_marketing/presentation/product_presentation/group0/79/79/81/91/98/19/4e/b2/x-cube-sbsfu_marketing-pres/files/x-cube-sbsfu_marketing-pres.pdf/jcr:content/translations/en.x-cube-sbsfu_marketing-pres.pdf

15 https://dspace.vsb.cz/bitstream/handle/10084/120984/1858-10404-1-PB.pdf?sequence=1

16 https://ieeexplore.ieee.org/document/7448561

17 http://www.provenrun.com/solutions/secure-firmware-update/

18 https://mkrak.org/2018/01/10/updating-embedded-linux-devices-part1/

19 https://www.timesys.com/security/software-firmware-update-design-considerations/

20 https://jumpnowtek.com/linux/An-upgrade-strategy-for-embedded-Linux-systems.html

21 https://events.static.linuxfound.org/sites/events/files/slides/elc16_angelatos.pdf