Test Automation 0.1.1.dev
Setup Guide

Version: 1.0

Revision History

S.N.	Change Description	Author(s)	Date
1.	Draft	Chetan Sharma, Harish Bansal	8th, September 2016
2.	Added Test server setup procedures for Ubuntu 14.04 and 16.04 new installation	Harish Bansal	7th February 2017

Table of Contents

Introduction

This release provides a driver testing framework and tool for running Driver tests on remote Boards. This guide describes procedure to setup Driver Testing Framework and tools in new environment.

Supported Platforms

Test Server	Any Linux Machine
Board	Standard/Custom Board

Test Server Setup

To start with Test Server setup download TestAutomation0.1 zip from Google Drive (Link: https://drive.google.com/open?id=0B3OaNpQczDSuc0VvWm5vczhLNWc**)**

Any Linux machine could be used as Test Server. However we have tested deployment on Ubuntu 16.04 and 14.04. Setup procedures for both these versions are mentioned beneath.

Required Python package Installation

Ubuntu 16.04

Execute following command on your Linux Test Server.

- Python-Pip Installation
  - ***$ sudo apt-get update***
  - ***$ sudo apt-get install libssl-dev libffi-dev python-dev***
  - ***$ sudo apt-get install python-pip***

- Install remaining python packages (paramiko, scp)
  - Go to {Extracted TestAutomation0.1}/automation directory
    - ***and Run command $ sudo -H pip install -r requirements.txt***

Ubuntu 14.04

Execute following command on your Linux Test Server.

- Python-Pip Installation

    - ***$ sudo apt-get update***
    - ***$ sudo apt-get install libssl-dev libffi-dev python-dev***
    - ***$ wget https://bootstrap.pypa.io/get-pip.py***
    - ***$ sudo python ./get-pip.py***
- Install remaining python packages (paramiko, scp)
    - Go to {Extracted TestAutomation0.1}/automation directory
        - ***and Run command $ sudo -H pip install -r requirements.txt***

Framework Installation

Below procedure is common for any Linux distribution.

- Download source from following link in section Framework Download Link on Test Server machine.
- Uncompress source code from compressed format.
    - ***$ unzip TestAutomation0.1.zip***
    - ***$ cd TestAutomation0.1/automation/dist***
- Goto automation/dist directory.
- Please execute following command to install installer on Test Server
    - ***$ sudo easy_install TestAutomationFramework-0.1.1.dev0-py2.7.egg***
- Make sure that installation is successful by verifying that executable file ts_test_runner is present under default directory after installation **/usr/local/bin**
    - *Run Command:* ***$ ts_test_runner***

SDK Creation

Timesys driver tests are written in C and shell by considering basic package of minimal embedded Linux operating system. We have ran these scripts on 3 different boards as TI Sitara AM335x Starter Kit, Boundary Devices Nitrogen 6 Quad Plus Max and BeagleBone Black (Open Source), however it can be setup and executed on any other board.

SDK Package Dependencies

For successfully building and running driver tests on board, test board BSP should have additional packages . These are

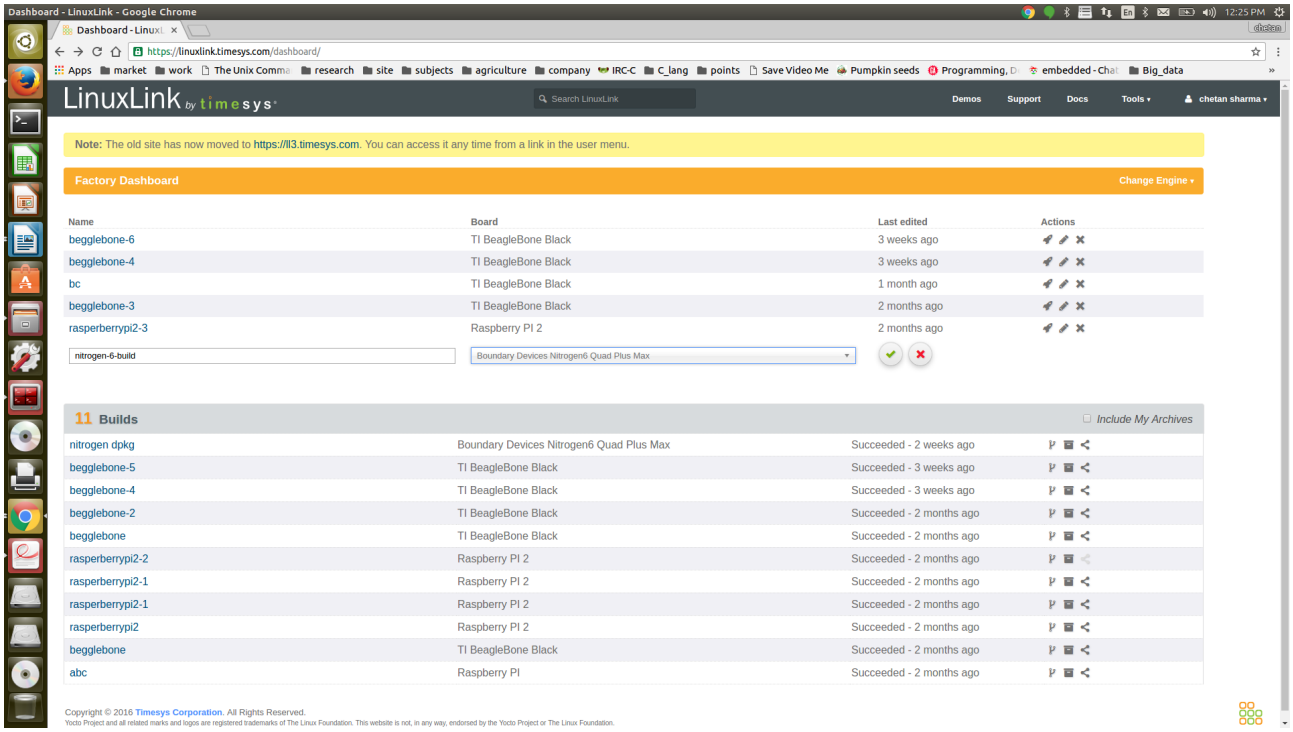
- **BusyBox** : It include basic Linux several basic Linux tools.
- **Bash** : It is require to execute shell scripts.
- **OpenSSH** : it is required to access Board remotely from another machine.
- **Make*** : It is necessary to compile and execute c code on board.

* Not required if you are cross-compiling test suites on test server machine using factory

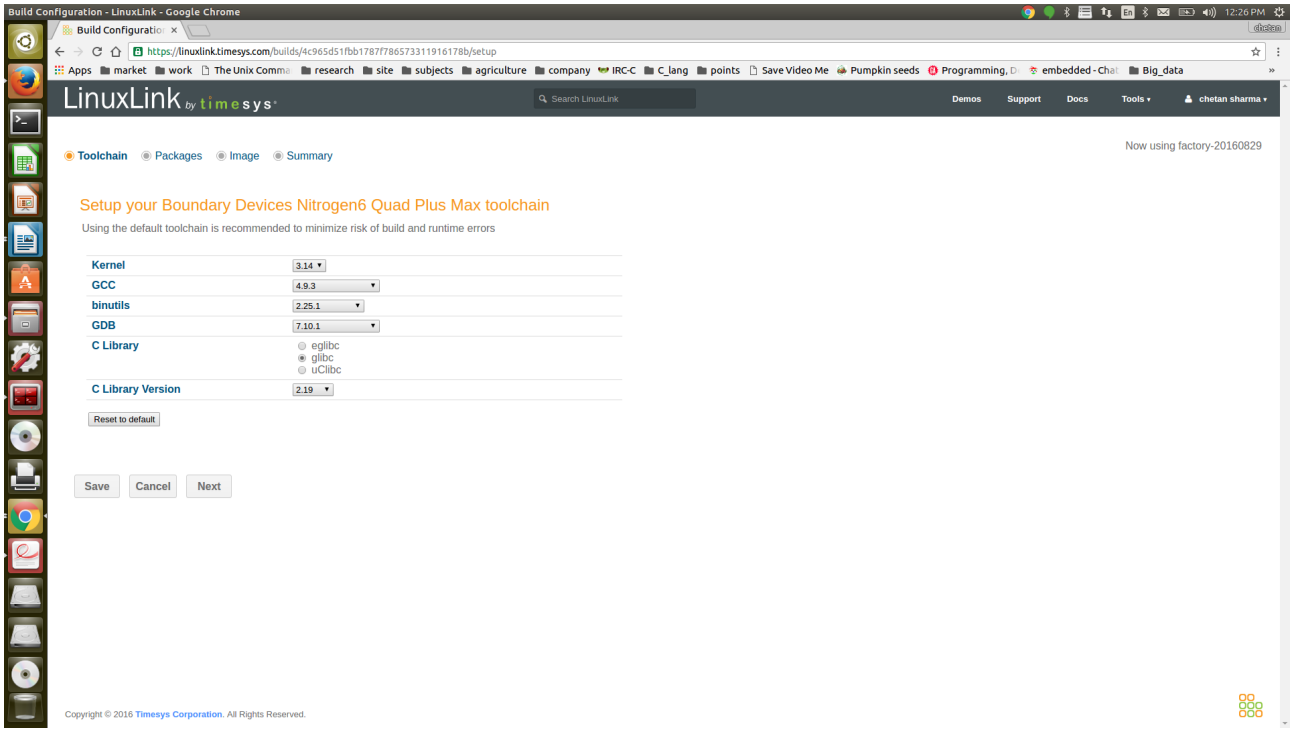
Building BSP with LinuxLink

These are the steps for building BSP with required dependent packages as described in section SDK Package Dependencies. Skip this section if BSP with required packages is already loaded onto board and you are cross-compiling test suites on Test Sever machine using factory (See section: Option II: Cross-Compile test suites on Test Server using Factory).

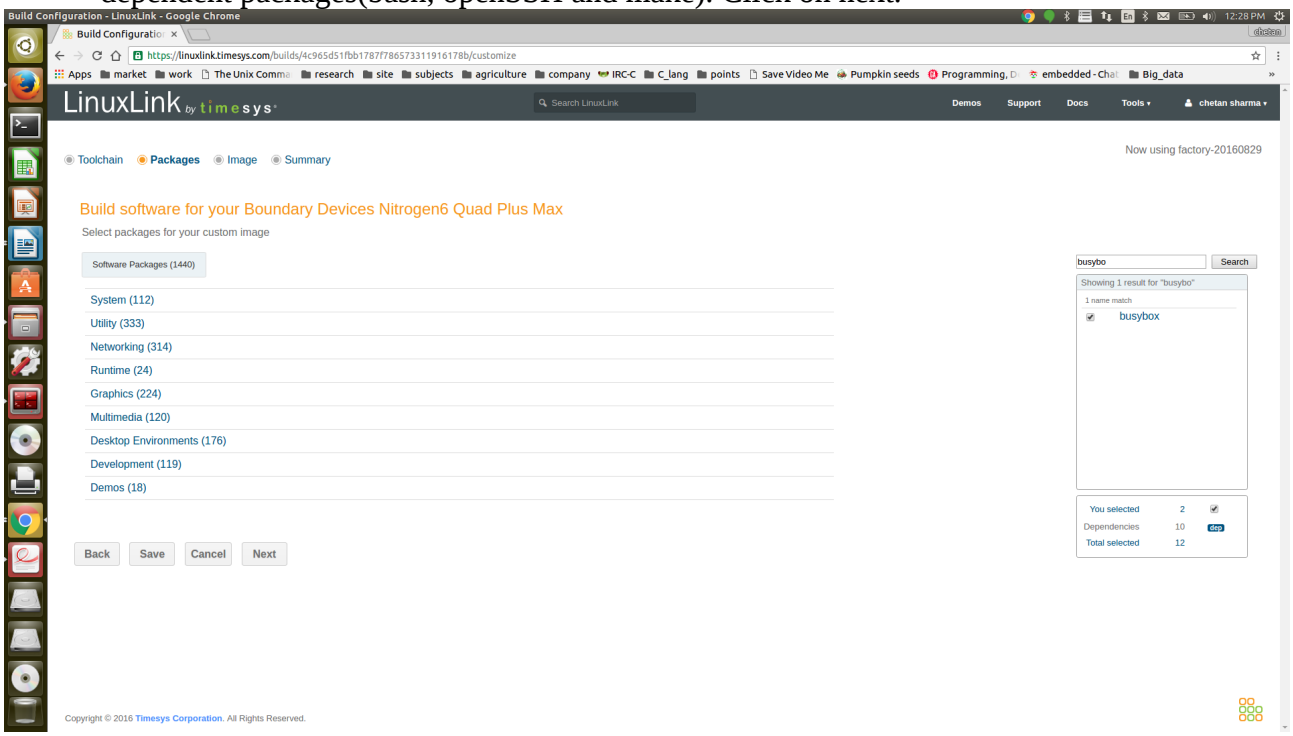
1) Login into LinuxLink using user credentials and select new options for build BSP.
2) Enter a name for your build and select your board from the drop down menu for example “Boundary Device Nitrogen6 Quad Plux Max”.



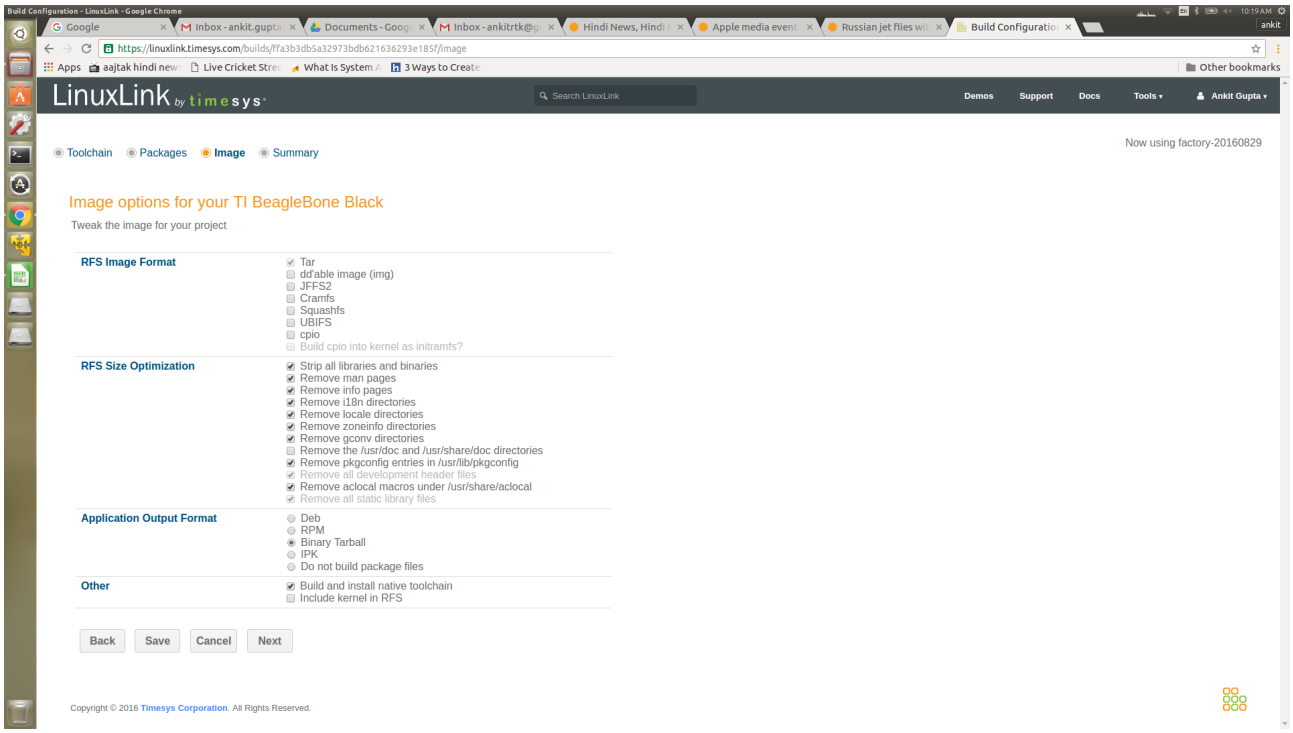
3) Then select your version of kernel, gcc, binutils, C-library etc. And click on next.



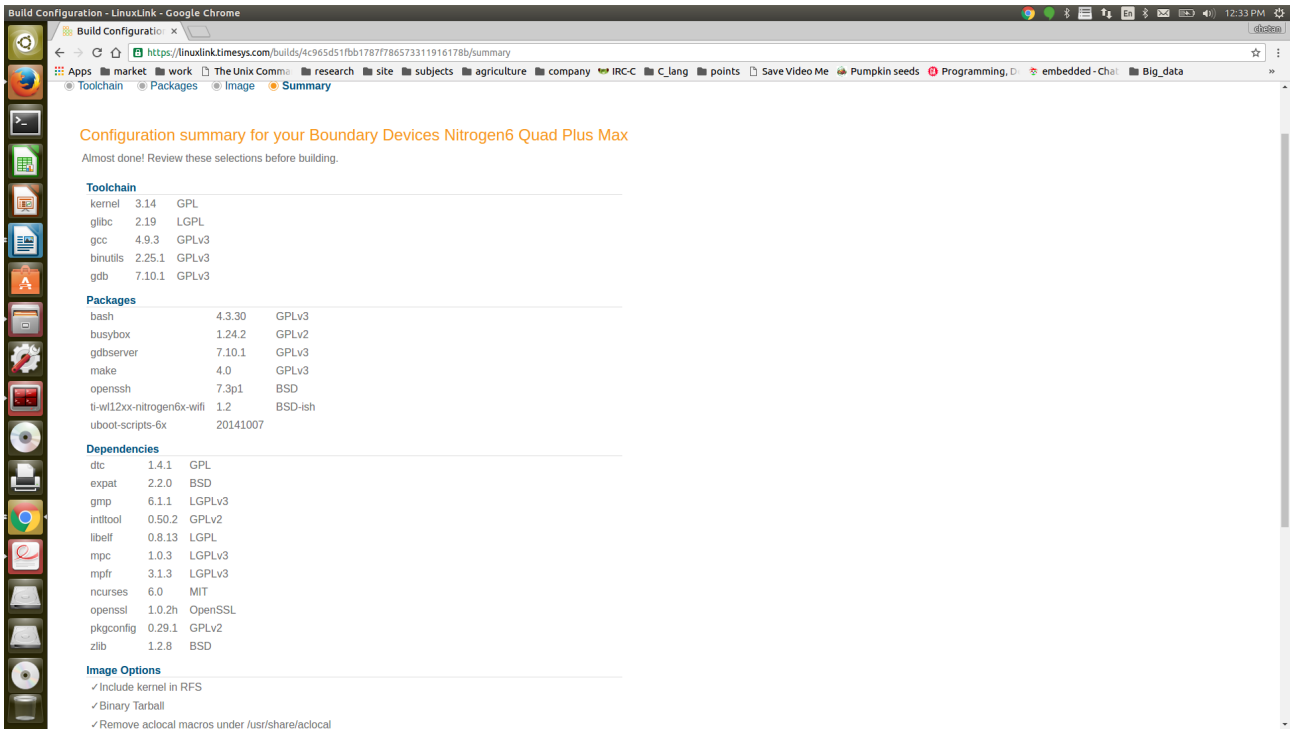
4) Select all dependent packages to include in your BSP, for busybox type busybox on to the right side of package search menu and select busybox from there and similarly for other dependent packages(bash, openSSH and make). Click on next.



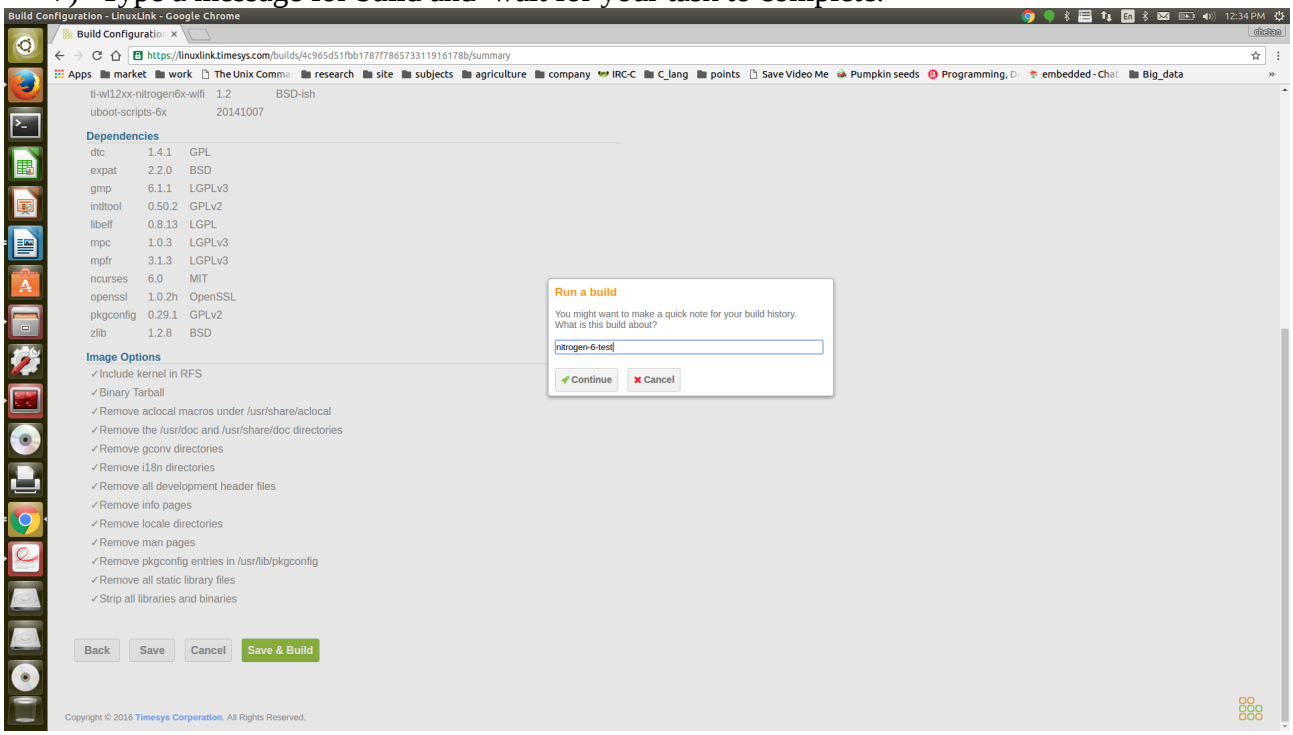
5) Check "*Build and install native toolchain*" (Its necessary for building test suites on board) and than click **Next**



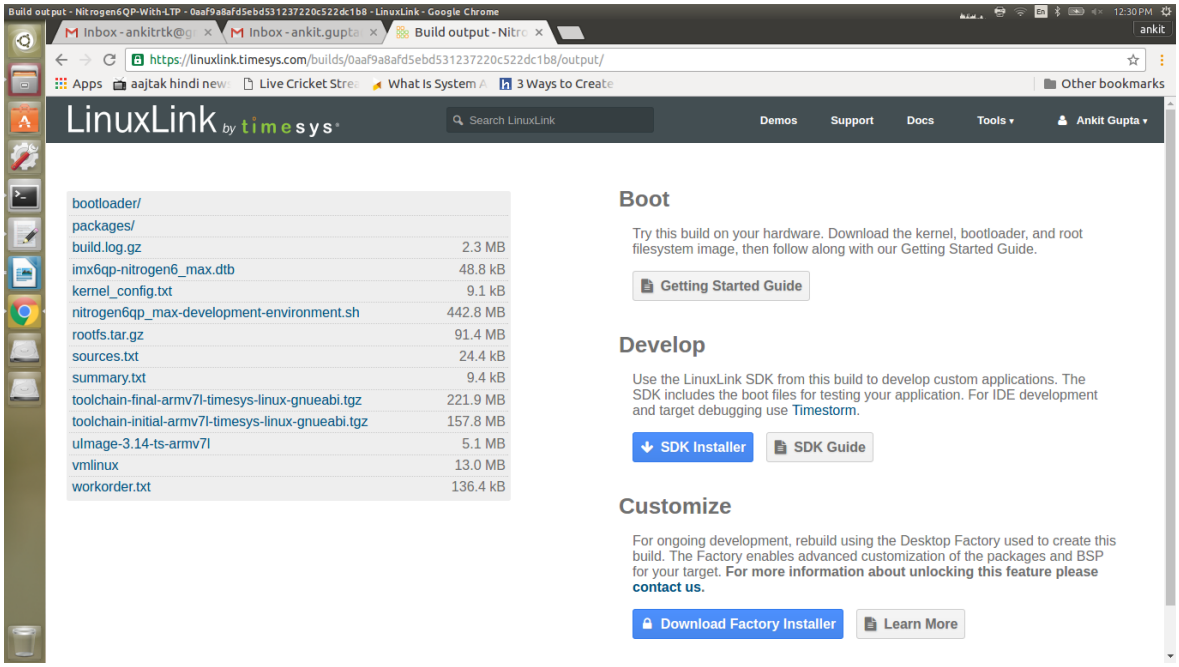
6) Verify the summary of your build i.e License, dependencies etc. before triggering the final build and click on save & build to proceed further.



7) Type a message for build and wait for your task to complete.



8) After this if your build is successful download the rootfs, uimage and bootloader and follow the below steps for board bring up from SD card.



Board Setup

Flash BSP on Board

You need to follow instructions from LinuxLink Documentation for flashing BSP on your board.
For two boards – Nitrogen 6 Quad plus max and BeagleBone Black you could follow below steps

Nitrogen 6 Quad Plus Max

- Partitioning the SD card

If you want to use a different SD card or its contents become corrupted, you can use the fdisk tool to create a single Linux partition on your SD card. Please note that all data on the card will be lost upon completion of these steps.

1. Unmount the partition if it was automounted by using the umount command.

$ umount /dev/sdX1

2. As root, run the fdisk utility on the drive.

$ sudo fdisk /dev/sdX

3. In fdisk, Delete the existing partition table and create a new one using the o command.
   Command (m for help): o
   Building a new DOS disklabel with disk identifier 0x8b025602.
   Changes will remain in memory only, until you decide to write them.
   After that, of course, the previous content won't be recoverable.
4. Create a new primary partition using the n command.
   Command (m for help): n
   Partition type:
   p primary (0 primary, 0 extended, 4 free)
   e extended
   Select (default p): p
   Partition number (1-4, default 1): 1
   First sector (2048-30679039, default 2048):
   Using default value 2048
   Last sector, +Nitrogen 6 sectors or +sizeK,M,G (2048-30679039, default 30679039):
   Using default value 30679039
5. Verify that the partition table is correct by using the p command. It should look similar to the following:
   Command (m for help): p
   Disk /dev/sdX: 15.7 GB, 15707668480 bytes
   64 heads, 32 sectors/track, 14980 cylinders, total 30679040 sectors
   *Units = sectors of 1 * 512 = 512 bytes*
   Sector size (logical/physical): 512 bytes / 512 bytes
   I/O size (minimum/optimal): 512 bytes / 512 bytes
   Disk identifier: 0x6eaae8f8

Device Boot start End Block Id System
/dev/sdX1 2048 30679039 83 Linux

6. This step will destroy all data on the SD Card - Write the partition table to the card using the w command.
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
Syncing disks.

7. Format the first partition of the SD card with the ext4 filesystem using the mkfs.ext4 tool.

$ sudo /sbin/mkfs.ext4 -L rfs /dev/sdX1

- Writing Boot Files to the SD Card
    1. Mount the partition. You can remove and reinsert the card to trigger the automount, or you can use the mount command to mount the partition to an arbitrary location.

$ sudo mount /dev/sdX1 /media/rfs

    2. As root, extract the rootfs.tar.gz archive to the mounted directory. This file is located at build_armv7l-timesys-linux-<libc>/images/rfs/ on Desktop Factory builds.

$ sudo tar xzf rootfs.tar.gz -C /media/rfs

    3. If you have included the kernel image in your rootfs, this next step is not necessary As root, create the boot directory on the RFS partition of the card.

$ sudo mkdir /media/rfs/boot

    4. As root, copy the uImage file, uImage, to the boot directory on the RFS partition of the card.

$ sudo cp uImage /media/rfs/boot/

    5. As root, umount the SD Card.

$ sync
$ sudo umount /media/rfs

    6. Remove the SD Card from the host machine, and insert it into the SD Card slot on the target board J18. You should hear the card 'click' into place.

- Boot Commands
    1) setenv mmc_init 'mmc dev 0'
    2) setenv ext4_load 'ext4load mmc 0:1 10800000 /boot/uImage; ext4load mmc 0:1 11000000 /boot/imx6qp-nitrogen6_max.dtb'
    3) setenv bootargs_mmc 'setenv bootargs console=ttymxc1,115200 root=/dev/mmcblk0p1 rw rootwait'
    4) setenv bootcmd_mmc 'run mmc_init; run ext4_load; run bootargs_mmc; bootm 10800000 - 11000000'
    5) setenv bootcmd 'run bootcmd_mmc'
    6) saveenv

BeagleBone Black (BBB)

1) Files Required
   - MLO
   - u-boot.img
   - uImage-4.1-ts-armv7l
   - beaglebone_black.dtb
   - rootfs.tar.gz

   When you build your BSP using LinuxLink, then after successfully build go to the build output and where you can download all the above listed required files. MLO and u-boot.img file is present in bootloader folder.
2) Partition the SD Card
   - umount the mounted SD Card.
   - As root, run the **fdisk** utility on the drive.

Device	Boot	Start	End	Blocks	Id	System
/dev/sdX1	*	2048	133119	65536	c	W95 FAT32 (LBA)
/dev/sdX2		133120	30679039	14773960	83	Linux

   - Format the partitions:
     - `sudo /sbin/mkfs.vfat -n boot /dev/sdX1`
     - `sudo /sbin/mkfs.ext4 -L rfs /dev/sdX2`
   - Mount the partitons to /media/boot and /media/rfs

3) Write Boot files to SD Card
   - sudo cp MLO /media/boot
   - sudo cp u-boot.img /media/boot
   - sudo cp uImage-4.1-ts-armv7l /media/boot
   - sudo cp beaglebone_black.dtb /media/boot
   - sudo tar -xzf rootfs.tar.gz -C /media/rfs
   - Unmount the partitions and insert the SD Card into the board.

4) Commands to Boot the Board
   - Hold down **S2**, then power up the board by removing power and reapplying it.
   - Stop at u-boot prompt by pressing enter.
   - Provide the below mentioned commands:
     - Set Environment Variables:
       - setenv bootargs console=ttyS0,115200 root=/dev/mmcblk0p2 rw rootwait
       - setenv load_kernel fatload mmc :1 0x82000000 uImage-4.1-ts-armv7l
       - setenv load_dtb fatload mmc :1 0x88000000 beaglebone_black.dtb
       - setenv bootcmd mmc rescan\; run load_kernel load_dtb\; bootm 0x82000000 - 0x88000000
       - saveenv
     - Load the Kernel

        - mmc rescan
        - fatload mmc :1 0x82000000 uImage-4.1-ts-armv7l
    - Load the Device Tree
        - mmc rescan
        - fatload mmc :1 0x88000000 beaglebone_black.dtb
    - Boot the Kernel
        - bootm 0x82 BeagleBone Black(BBB)000000 - 0x88000000

Network connectivity Setup

Steps to connect Test Server and Test Board.

- Connect Test Server and Test Board using P2P connection or through INTERNET.
- After connection verify that IP address is assigned automatically through DHCP to both ends and perform ping test from Test Server to Test Board.
- Verify SSH Connection from Test Server to Test Board.

Option I: P2P Connection

Test Server

Steps required to perform on Test Server for P2P connection. *These steps were performed and Tested on Ubuntu 16.04, for other Linux distributions these steps may vary.*

- Connect LAN Ethernet cable between Test Server and Test Board.
- Goto "**Network Connections**" and select Ethernet Device usually it is named with Connection name "**eth0**".
- Go to "I**Pv4 Settings**" tab and select "**Method**" as "**shared to other computers**" from drop down list.

Test Board

Steps required to perform on Test Board for P2P connection.

- Login to Test Board.
- Enable DHCP on Test Board, so that device can get IP Address from Test Server.
    - Write these lines in **/etc/network/interfaces** file by editing using text editor(vi).

      auto eth0
      iface eth0 inet dhcp
    - Perform Reboot Operation on Test Board to update this change.

        - Run Command: ***$ sync***
        - Run Command: ***$ reboot***

Option II: Intranet Connection

If test board is not directly connected with the Test Server, but accessible through SSH, perform following tests for verifying Test Server and Test Board connectivity.

Ping Test Steps

- Gather IP address of Test Board using following command.

$ **ifconfig**

eth0 Link encap:Ethernet HWaddr 68:9e:19:57:6e:c3
*inet addr:**192.168.7.2** Bcast:192.168.7.3 Mask:255.255.255.252*
UP BROADCAST MULTICAST MTU:1500 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
Interrupt:40

- Start ping test from Test Server using following command.

timesys@timesys-GL552VW:~$ ping 192.168.7.2

PING 192.168.7.2 56(84) bytes of data.
64 bytes from (192.168.7.2): icmp_seq=1 ttl=47 time=886 ms
--- 192.168.7.2 ping statistics ---
2 packets transmitted, 1 received, 50% packet loss, time 999ms

SSH Connection Test Steps

From Test Server do ssh to Test Board by using existing user credentials of Test Board to verify ssh connectivity. Please use following command as reference to test ssh connection. In this command we take root as user to login to Test Board.

**timesys@timesys-GL552VW:~$ ssh root@192.168.7.2*

* Board's root password should be set before running this command. You may set root password on board by connecting it with Minicom and running "$passwd root"

Test suites Compilation

Option I: Compile test suites on Board

Test Board

- Download source from following link in section Framework Download Link on any machine.
- Uncompress source code from compressed format.
  - ***$ unzip TestAutomation0.1.zip***
  - ***$ cd TestAutomation0.1***
- Copy **timesys-test-project** to Test Board by using scp Linux tool or FTP Linux tool.
  - ****$ scp -r timesys-test-project root@<test board IP Address>:/opt***

** Board's root password should be set before running this command. You may set root password on board by connecting it with Minicom and running "$passwd root"*

- In Test Board goto timesys-test-project directory and execute following command to compile source code
  - ***$ ssh root@<test board IP Address>***
  - ***$ cd /opt***
  - ***$ make***
- Make sure this compilation will not fail and Perform following command on successful compilation of source code.
  - ***$make install***
- Verify that **/usr/tstp** directory is created and all binary is copied to this directory. Perform directory list operation to verify.

  $ ls -al /usr/tstp
  -rw-r--r-- 1 root root 105 Aug 30 2016 automated.conf
  drwxr-xr-x 25 root root 4096 Aug 30 2016 bin
  -rwxr-xr-x 1 root root 3062 Aug 30 2016 Test.sh

Option II: Cross-Compile test suites on Test Server using Factory

- Grab a PC having Ubuntu64 bit.

- Install following program on Ubuntu
  $ sudo apt-get install git-core gperf libesd0-dev libwxgtk3.0-dev build-essential
  $ sudo apt-get install libx11-dev libncurses5-dev g++-multilib lib32z1 flex
  $ sudo apt-get install libreadline-dev libstdc++5 tofrodos zlib1g-dev bison
  $ sudo dpkg-reconfigure dash ("Selection 'NO'")

- Clone Desktop Factory from timesys git link git://engservices.timesys.com/factory.git .

- Go to the factory working directory.

  $ cd factory/

- Select architecture board and toolchain configuration using following instructions.
  $make menuconfig
  Select Target Configuration-> Target Architecture Options-> Board type
  Select Target Configuration-> Target Architecture Options-> Target Architecture
  Select Toolchain Configuration-> Toolchain (Build)

For example:- Select these configuration for BeagleBone board
Select Target Configuration-> Target Architecture Options-> Board (TI BeagleBone Black)
Select Target Configuration-> Target Architecture Options-> Target Architecture (arm)

Select Toolchain Configuration-> Toolchain (Build)

- Select the target package format of timesys-test-project (Eg :- tarball, deb package,etc) For tarball format follow these instructions

Select Target Configuration-> Output Package Format-> Select Binary tarball
Save changes in menuconfig and exit from it.

- Compile and fetch toolchain from provided path in above configuration using following instruction.

$ make clean
$ make toolchain

- Now run the following command to add the package to our desktop factory.

$ ./bin/new_pkg timesys-test-project Utilities skip external

This command will create the .config and project .mk files required for building the project.

- Now create a project source directory in the factory in the src/external/project-name

$ mkdir -p src/external/timesys-test-project/

- Copy the source files of the timesys test project downloaded from link provided in Setup Guide to the src/external/project-name

$ cp -rL <PATH_TO_PROJECT>/timesys-test-project/* src/external/timesys-test-project/

- Now select the package from the menuconfig or by the following command

$ make timesys-test-project-select

- Finally build the package and make

$ make timesys-test-project-package

- Find binary tarball package in build_xxxx/packages with prefix name as timesys-test-project.

$ **ls build_armv7l-timesys-linux-gnueabihf/packages/timesys-test-project-1.0-armv7l-timesys-linux-gnueabihf-1.tgz**

- Copy **timesys-test-project** to Test Board by using scp Linux tool or FTP Linux tool.

$ scp -r build_armv7l-timesys-linux-gnueabihf/packages/timesys-test-project-1.0-armv7l-timesys-linux-gnueabihf-1.tgz root@<test board IP Address>:/opt

- In **Test Board** goto timesys-test-project directory and uncompress files from targz compress format and copy to (/usr) target directory.

  $ ssh root@<test board IP Address>
  $ cd /opt
  tar -zxvf timesys-test-project-1.0-armv7l-timesys-linux-gnueabihf-1.tgz
  $ cp -r usr/* /usr/

- Verify that **/usr/tstp** directory is created and all binary is copied to this directory.
  Perform directory list operation to verify.

  $ ls -al /usr/tstp
  -rw-r--r-- 1 root root 105 Aug 30 2016 automated.conf
  drwxr-xr-x 25 root root 4096 Aug 30 2016 bin
  -rwxr-xr-x 1 root root 3062 Aug 30 2016 Test.sh

Appendix

Useful Links

Framework Download Link: https://drive.google.com/open?id=0B3OaNpQczDSuc0VvWm5vczhLNWc

Linux Link:

https://ll5.timesys.com/

Factory Installation:

https://linuxlink.timesys.com/docs/

Factory Git clone URL: git://engservices.timesys.com/factory.git