Firmware User Manual
iMX8M THOR96 Reference Design

Version	ProdV2.3
Status	Baseline
Date	28-Nov-2019

Contents

Figures

Tables

DOCUMENT DETAILS

➢ Document History

Version	Author		Reviewer		Approver	
	Name	Date (DD-MM-YYYY)	Name	Date (DD-MM-YYYY)	Name	Date (DD-MM-YYYY)

Table 1 : Documents History

Version	Description Of Changes
0.1	*initial draft*
0.2	Added new features as mentioned in release note 0.2
0.3	Added QT demos and new features as mentioned in release note 0.3
0.4	Added Alexa Demo
Beta1.0	Added Beta related changes
Beta1.1	Added steps to build Yocto image
Prod2.0	Added A2B Demo
Prod2.1	Update QT based demos
Prod2.2	Added updated ML demos in home folder
Prod2.3	Integrated A2B source code

Table 2 : Description of Changes

➢ Definition, Acronyms and Abbreviations

Definition/Acronym/Abbreviation	Description
Cd	Change directory
scp	Secure copy over the network
Dfl	Default
Wi-Fi	Wireless fidelity
LTE	Long-Term Evolution
BLE	Bluetooth low energy device
DSI	Display Serial Interface
CSI	Camera Serial Interface
A2B	Automotive Audio Bus

Table 3 : Definition, Acronyms and Abbreviations

➢ References

No.	Document	Version	Remarks
1	Release note V2.2	2.2	Integrated A2B source Code
2	Release note V2.1	2.1	Meta Layer Release v2.1
3	Alexa_User_Guide_v0.1	0.1	Alexa User Guide
4	FlashDocument.pdf	-	MGM111 Zigbee module flashing steps

Table 4 : References

Introduction

➢ Purpose of the document

- Purpose of this document is to use/understand/flash/demonstrate interfaces on iMX8M-THOR96 PLATFORM firmware.

➢ About the System

- This system contains iMX8M reference design with multiple interfaces, can be used for Human-machine interface experience.

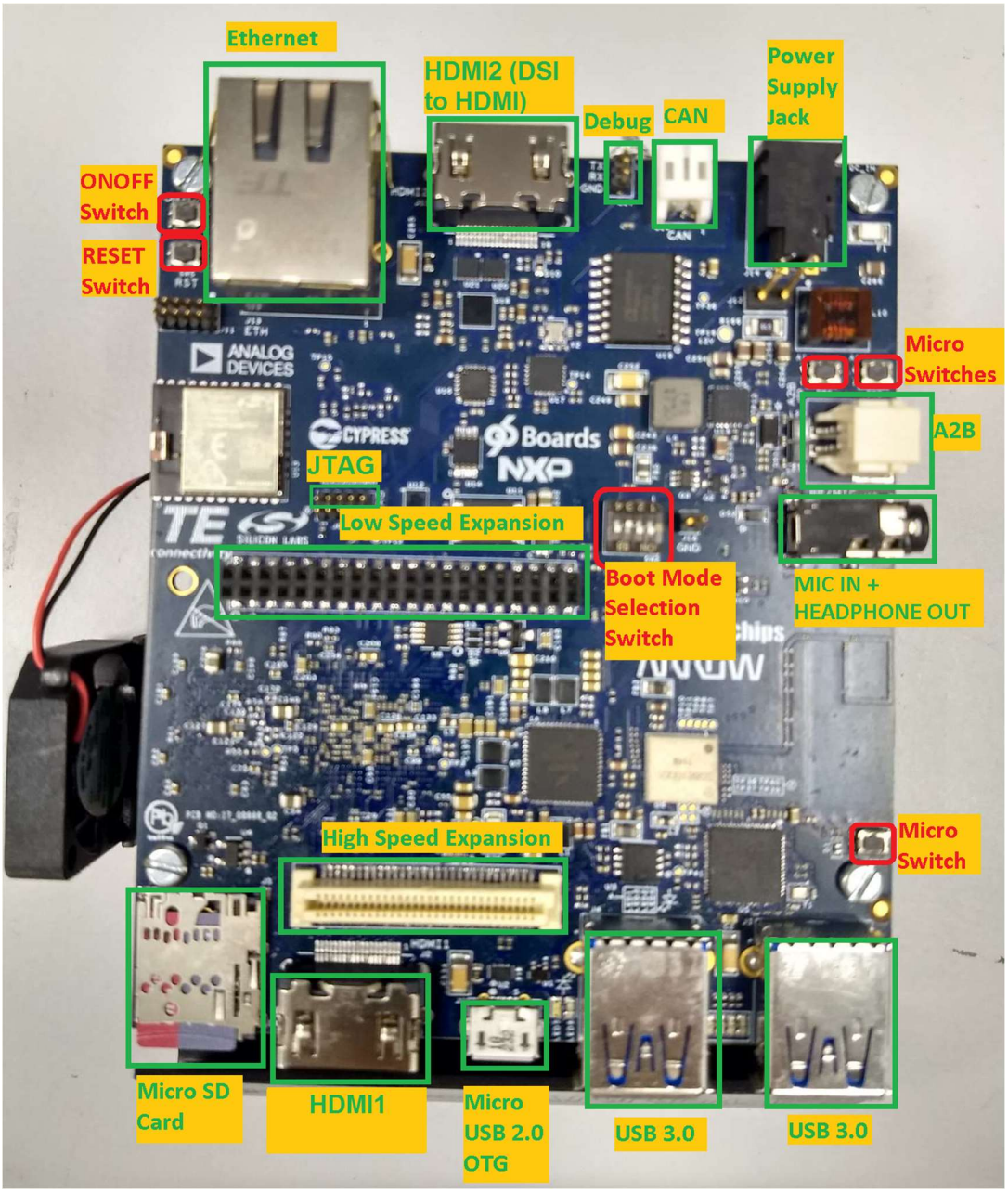

Figure 1 : iMX8M Thor96 Platform Connectors

➢ Before You Start

- Ensure you have x86 host system having Linux Ubuntu 16.04 LTS installed
- Basic understanding of Linux commands

REMARK: In case after the repo sync is completed and during the build process, if you see the error related to Vulcan please apply the patch using the file embedded here. This will help to remove the vkmark errors. Start the build process thereafter as mentioned.


Package

➢ Steps to build Yocto Image

We already prepared Meta-layer, which contains all the packages and BSP changes required for Thor96 firmware image. User need to download meta-layer first to build image for Thor96.

To build Thor96 firmware on LINUX HOST PC, user need to follow below steps:

- Open command prompt (CTRL+ALT+T) and install required packages to build.
  $: sudo apt-get install gcc g++ gawk wget git-core diffstat unzip texinfo gcc-multilib build-essential chrpath socat libsdl1.2-dev libsdl1.2-dev xterm sed cvs subversion coreutils texi2html docbook-utils python3-pip python-pip python-pysqlite2 help2man make desktop-file-utils libgl1-mesa-dev libglu1-mesa-dev mercurial autoconf automake groff curl lzop asciidoc u-boot-tools

- Now we need to download new repo for Thor96. Currently we are using kernel version 4.14.78-ga release repo.
- To download repo, first we need repo utility. For that need to follow below steps:
  $: mkdir ~/bin
  $: curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
  $: chmod a+x ~/bin/repo
  $: PATH=${PATH}:~/bin
- Now Download Yocto Project environment into local directory
  $: mkdir thor96-yocto-bsp
  $: cd thor96-yocto-bsp
  $: repo init -u https://source.codeaurora.org/external/imx/imx-manifest -b imx-linux-sumo -m imx-4.14.78-1.0.0_machinelearning.xml
  $: repo sync
- Above **repo sync** command will download default NXP source code into our local directory "**thor96-yocto-bsp**".
- Download meta-einfochips-*.tar.gz package and extract it. You will get **"meta-einfochips" meta-layer folder.**
- Now we need to copy our meta-layer "**meta-einfochips**" into "**sources**" folder.
- "**Meta-einfochips**" contains source code for our both the custom boards AIML and THOR96.
- Therefore, we need to setup our environment based on target machine (board).
- For that we need to follow below steps: (Setup Yocto Build)

Apply patch **"0001-iMX8-Added-required-BBLAYERs-in-local-bblayer.conf.patch"**
$: cd sources/meta-fsl-bsp-release
$: git apply ../meta-einfochips/conf/0001-iMX8-Added-required-BBLAYERs-in-local-bblayer.conf.patch
In above patch we add three new BBLAYERs to our local **bblayers.conf (i.e. meta-webserver, meta-einfochips, meta-imx-machinelearning)**

Now setup build environments for Thor96 board.
$: cd ../../
$: EULA=1 MACHINE=imx8mqthor96 DISTRO=fsl-imx-xwayland source ./fsl-setup-release.sh -b bld-xwayland-thor96

- After successful setup, we got new build directory **bld-xwayland-thor96,** if it is not there previously. Make sure in build directory "**conf/bblayers.conf"** contains all our required BBLAYERs (which we added through patch).
- Now we are good to go and can build image.

  $: bitbake fsl-image-qt5

- If user want to clean previously build image and want to run it again then we must first clean it with command **"cleanall" or "cleansstate"**
  $: bitbake fsl-image-qt5 -c cleanall

  $: bitbake -v fsl-image-qt5 (If user want to turn on verbose)

- If user want to clean any particular package then also we can do that with command **"cleanall" or "cleansstate"**
  $: bitbake <PACKAGE_NAME> -c cleanall
  $: bitbake <PACKAGE_NAME>

  e.g.
  $: bitbake linux-imx -c cleanall
  $: bitbake linux-imx (Build linux kernel only)

  Same way
  $: bitbake u-boot-imx -c cleanall
  $: bitbake u-boot-imx (Build uboot code only)

  $: bitbake imx-gpu-sdk -c cleanall
  $: bitbake imx-gpu-sdk (Build gpu sdk only)

  $: bitbake opencv -c cleanall
  $: bitbake opencv (Build opencv package)

  $: bitbake python3-numpy -c cleanall
  $: bitbake python3-numpy (Build numpy python package for python3)

- Please note that, if you re-build any module then it is better to re-build all modules, which are dependent on that module. For example, if you change anything in Linux kernel code and rebuild it using above commands then you must need to re-build kernel-module-laird, imx-gpu-sdk etc. packages to avoid conflicts.
- After successful build final sd card image reside at below location:
  thor96-yocto-bsp/bld-xwayland-thor96/tmp/deploy/images/imx8mqthor96/
  Filename should be **fsl-image-qt5-imx8mqthor96.sdcard.bz2** which is soft link of original build image file **fsl-image-qt5-imx8mqthor96-<TIMESTAMP>.rootfs.sdcard.bz2**

➢ Get the firmware package

- Download the provided SD card (sdcad.bz2) image in Linux pc
- Open terminal in host pc from left desktop panel or using keyboard shortcut (**ctrl + alt + t)**
- From command terminal traverse the location where downloaded firmware image is residing using **cd** command
  $: cd /home/user/download/imximages/
- use **ls** command to verify the existence of downloaded image
  $: ls -l
- Verify md5 check sum of downloaded image with given md5sum.
  $: md5sum <image name>.sdcard.bz2
- Extract the provided **.bz2** image using **bunzip2** command, which will take couple of minutes.
  # bunzip2 -dkf <image_name>.sdcard.bz2
- Once done, will end with **.sdcard** image in the same directory and can again be verified using **ls -l** command.

➢ Flash the firmware image to SD Card in LINUX HOST PC

- Plugin micro SD card into x86 Linux Host PC
- *Verify the node created for SD card into /dev directory*
  # ls -l /dev/sd*
- Open terminal and traverse to the location where downloaded firmware image is residing using **cd** command
- Ensure the extracted firmware image's file format is **.sdcard** using **ls -l** command
- Use below command for flashing if the SD card's entry in Linux is **/dev/sdb**
  # sudo dd if=<image_name>.sdcard of=/dev/sdb bs=1M conv=fsync ;sync
- Above command will take couple of minutes or more (depending upon PC config) to flash the SD card
- Once done remove and insert the SD card, two drives will get mounted if the above command is successful, named <boot> and <rootfs>
- **Eject (safely remove) SD card from host PC and plug it into board's SD card slot**

➢ Flash the firmware image to SD Card in Windows HOST PC

- Plugin micro SD card into x86 Windows Host PC
- Install win32 Disk Imager (*https://sourceforge.net/projects/win32diskimager/*)
- Format SD card with **FAT** file system.
- Plug SD card with card reader. It must shows any drive like **"E:"**

- Download appropriate production image ***.sdcard.bz2**
- Extract ***.sdcard.bz2** image using winzip or 7-zip. It will create ***.sdcard** image.
- Run Win32 Disk Imager
- Select **.sdcard** image file and target drive i.e. **E:** for input Image File. (see below figure)

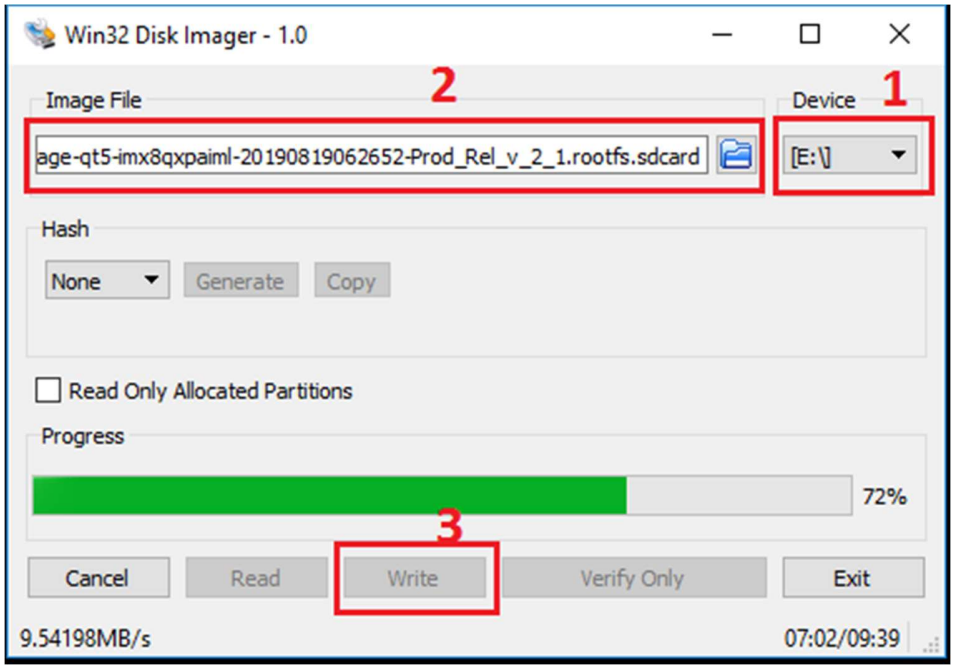

Figure 2: Win32 Disk Imager for flashing SD Card

- Click on **Write**.
- After successful transfer, success message will pop up and we got around 63.8 MB FAT partition. Inside that, we get different dtb files and Image.

➢ Hardware Installation

- Place hardware board on statically clean place
- Plug flashed SD card to J5 SD card slot.
- Plug serial cable's micro end to board's J10 Connector (near Ethernet connector) and USB end to host x86 pc's usb connector.
- Plug Ethernet cable to board's Ethernet connector J12.
- Apply 12V-5A power supply (provided with board) to board on J14 **DC_IN** connector. After all the other hardware setup is done and required interfaces are connected to board.

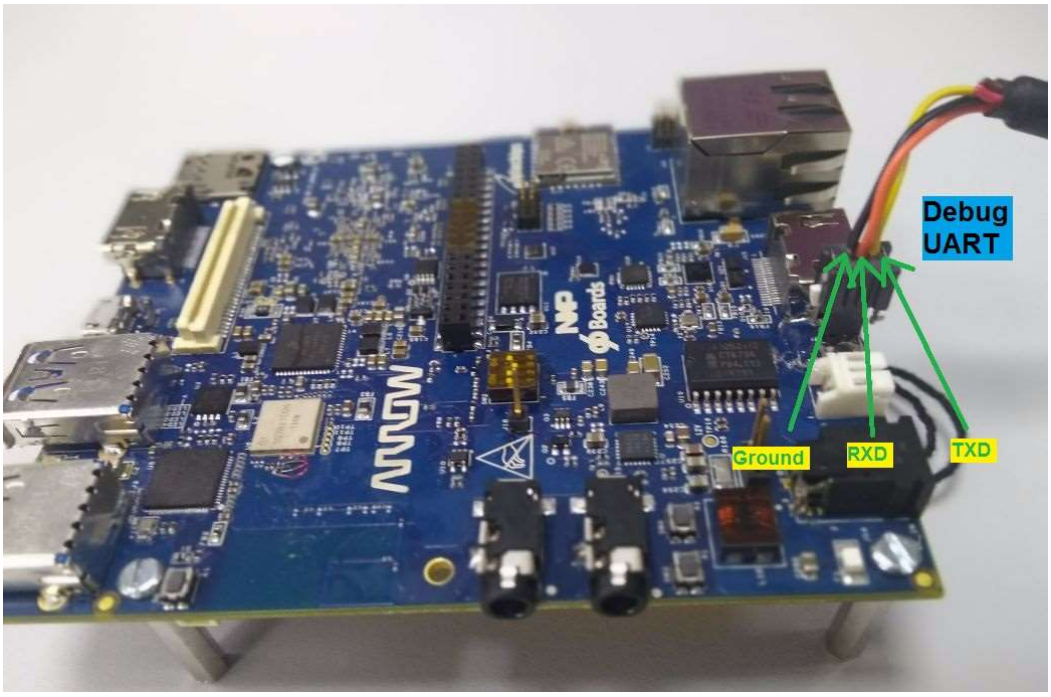

Figure 3 : Thor96 Platform UART Connections

➢ Open board's terminal- console (minicom) on x86 host pc

- Ensure SD card is flashed and serial cable is plugged in into board as per mentioned in hardware setup.
- Attached serial cable's USB end to host x86 PC's USB.
- Ensure **minicom** is installed in x86 Ubuntu pc
- Apply below command to open serial command's setting.
  $: sudo minicom -s
- **set baud rate and other setting as per below**
  - baud rate=115200,
  - parity=none
  - hardware flow control = none
  - software flow control = none
  - serial device= /dev/ttyUSB0
  - save setup as dfl
- Once board gets power-up, above configured terminal will show logs on x86 and can interact with board using this open terminal

RUNNING-DEMOS

1. Ethernet demo

- Plug in Ethernet cable to target board as per above figure.
- Power up the board.
- Once board gets booted, apply below command using console (minicom require)
  # ifconfig.
  # ping <any server ip>

2. HDMI Demo

- Ensure board is not powered up and SD card is flashed with the latest provided image.
- Insert HDMI cable into board's J2 HDMI connector.
- Apply power to board and go to terminal of x86 host system and open board's console as mentioned above
- Hold boot on u-boot screen by pressing any key on host machine keyboard(immediate after boot within 3 seconds)
- Apply dtb file changes as per below command on u-boot console
  # setenv fdt_file fsl-imx8mq-thor96.dtb
  # saveenv
  # boot
- Console will show booting logs
- Once booting is completed, console will hold on login prompt where user can enter username as **root**. (no password)
- At this time connected HDMI display will show grey image of desktop and should stop complaining about "No Signal"

Play Video Test pattern on HDMI display

- Go to board's console and type below command from x86 minicom
  # gst-launch-1.0 videotestsrc ! autovideosink
- Above command will show color strips on HDMI display

Play local videos on HDMI display with audio

- Ensure Ethernet is connected with board
- Go to board's console and type below command from x86 minicom
  # ifconfig
- Get the ip address of Ethernet eth0 interface and note down.
- Go to x86 host system and download sample mp4 video with audio.
- Locate to video location from command line in x86 (no minicom require)
- Apply below command

$: scp ./Sample_Video_with_audio.mp4 root@<noted ip address of board>:/home/root/

- This will copy the video file from host x86 to board's /home/root location
- Go to board's console (require minicom) and ensure video got copied using ls -l command, will show you Sample_Video_with_audio.mp4 in current directory.
- Be in the board's console and apply below command to play video over HDMI Display with audio (HDMI Display should have support of audio)

  # gst-launch-1.0 filesrc location=/home/root/Sample_Video_with_audio.mp4 ! decodebin name=dec ! videoconvert ! autovideosink dec. ! audioconvert ! audioresample ! alsasink device=plughw:0,0

  Above command will print logs on console of board and will be played over HDMI display with audio. In above command if “alsasink device =plughw:0,0 “ then audio will play on audio codec(J10) and if “alsasink device =plughw:4,0” then audio will play on HDMI Display device.

  Or

  # gplay-1.0 Sample_Video_with_audio.mp4

  Here, to play audio successful user must need to have correct hardware entry inside **/etc/asound.conf**. If **hardware 0** is specified then audio will be played on **SAI** audio jack and if **hardware 3** is specified then audio will be played on **HDMI**. By default, audio will be played over HDMI.

3. Dual Display Demo

- Ensure board is not powered up and SD card is flashed with the latest provided image.
- Insert one HDMI cable into board's J2 HDMI connector, another HDMI cable to HDMI2 on j15 connector
- Apply power to board and go to terminal of x86 host system and open board’s console as mentioned above
- Apply dtb file changes as per below command on u-boot console
  # setenv fdt_file fsl-imx8mq-thor96-dual-display-b3.dtb
  # saveenv
  # boot
- Console will show booting logs
- Login to board using root username with no password.

Play local videos on HDMI display (Dual)

- Copy local sample_video.mp4 & sample_video2.mp4 videos using **scp** command to on board sd card as mentioned in above demo.
- Go to board's console (require minicom) and ensure video got copied using ls -l command, will show you sample_video.mp4 in current directory.
- Apply below command to play video over HDMI(j2) Display
  # gst-launch-1.0 filesrc location=sample_video.mp4 typefind=true ! video/quicktime ! qtdemux ! queue max-size-time=0 ! vpudec ! queue max-size-time=0 ! kmssink sync=true &
- With not much delaying apply below command to play video over HDMI2 (j15) Display
  # gst-launch-1.0 -v filesrc location=sample_video2.mp4 typefind=true ! video/quicktime ! aiurdemux ! queue max-size-time=0 ! vpudec ! waylandsink &
- The both the connected HDMI will show dual video demo playback.

Create RTSP network for network stream testing

- First Connect Thor96 board with linux PC where we created RTSP network.
- We first create static network in linux PC.
- For static IP, edit network (Ethernet) connection as given figures.

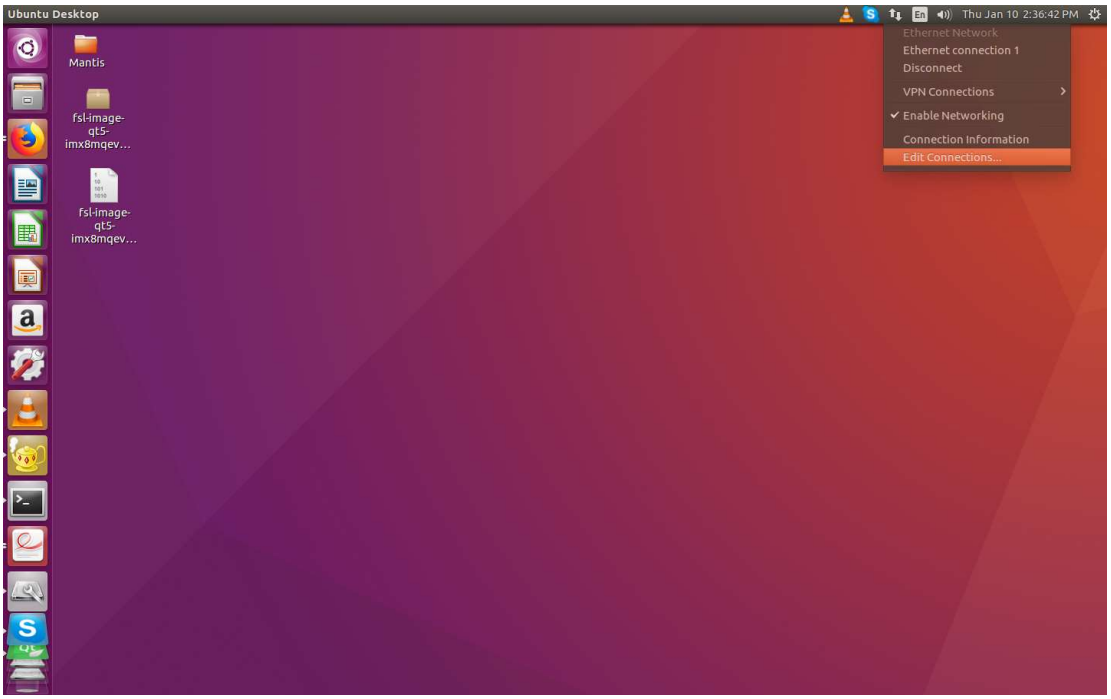

Figure 4 : Edit network connection

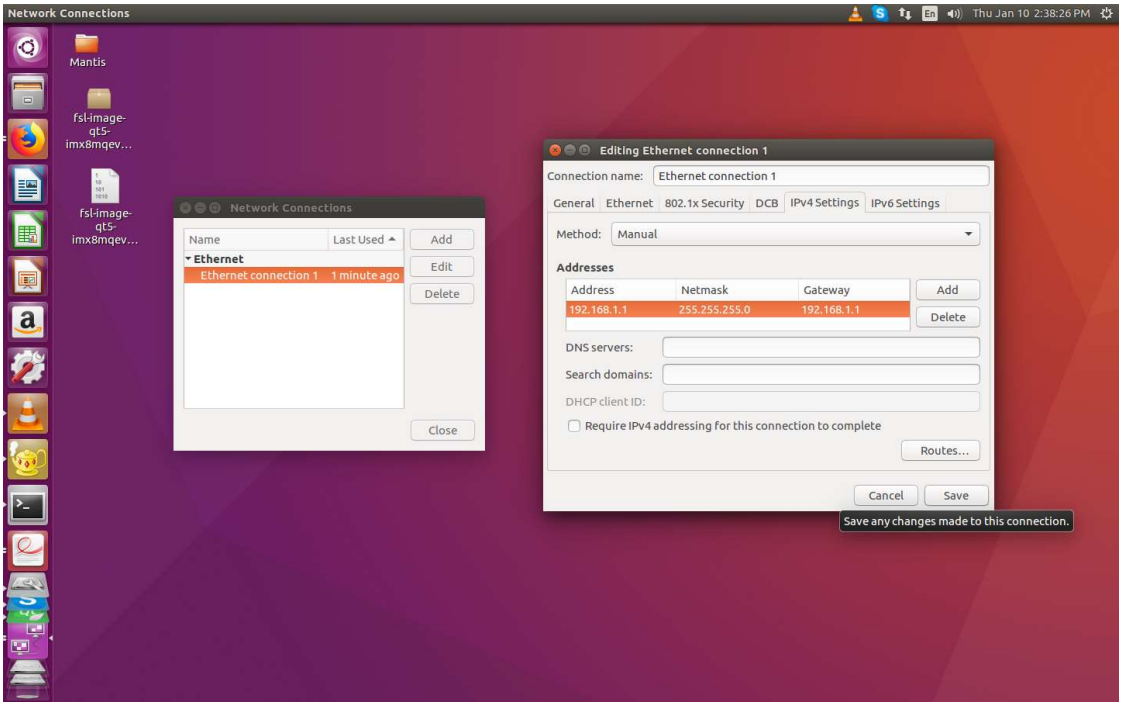

Figure 5 : Create Static network

- Here we **edit** Ethernet connection and change **IPV4 setting to "manual".**
- Set static IP and gateway to 192.168.1.1 with netmask 255.255.255.0.
- Now we setup RTSP stream server on Linux PC as per below images.
- Open VLC media player and start network stream.

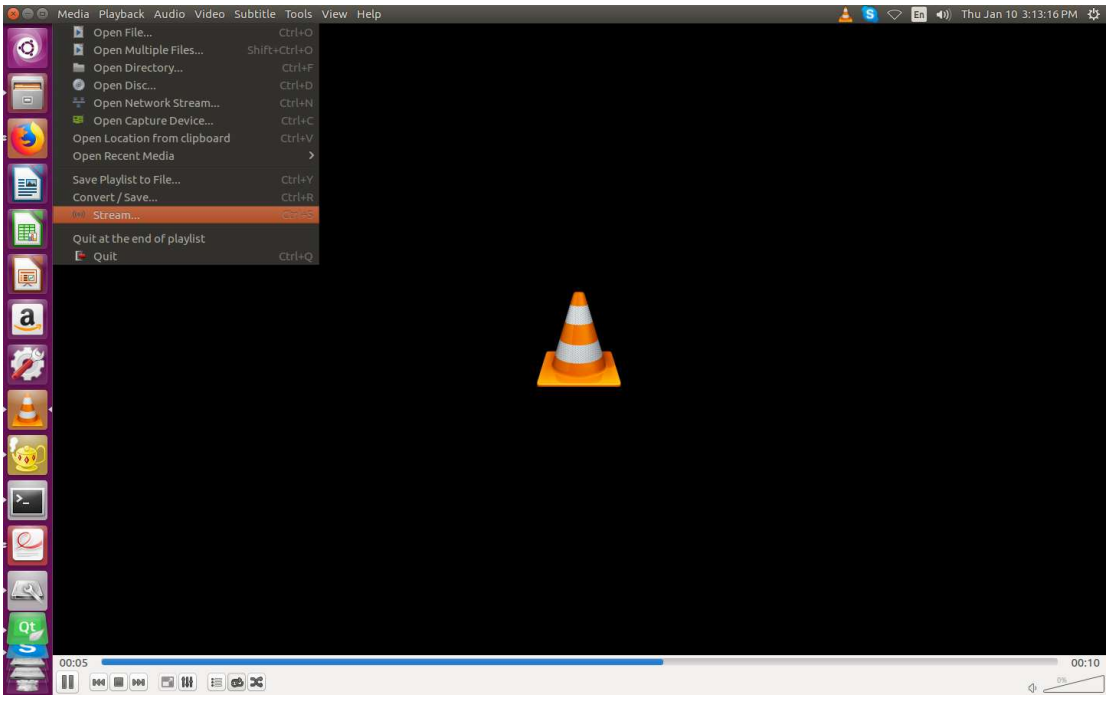

Figure 6 : Start Network Stream in VLC

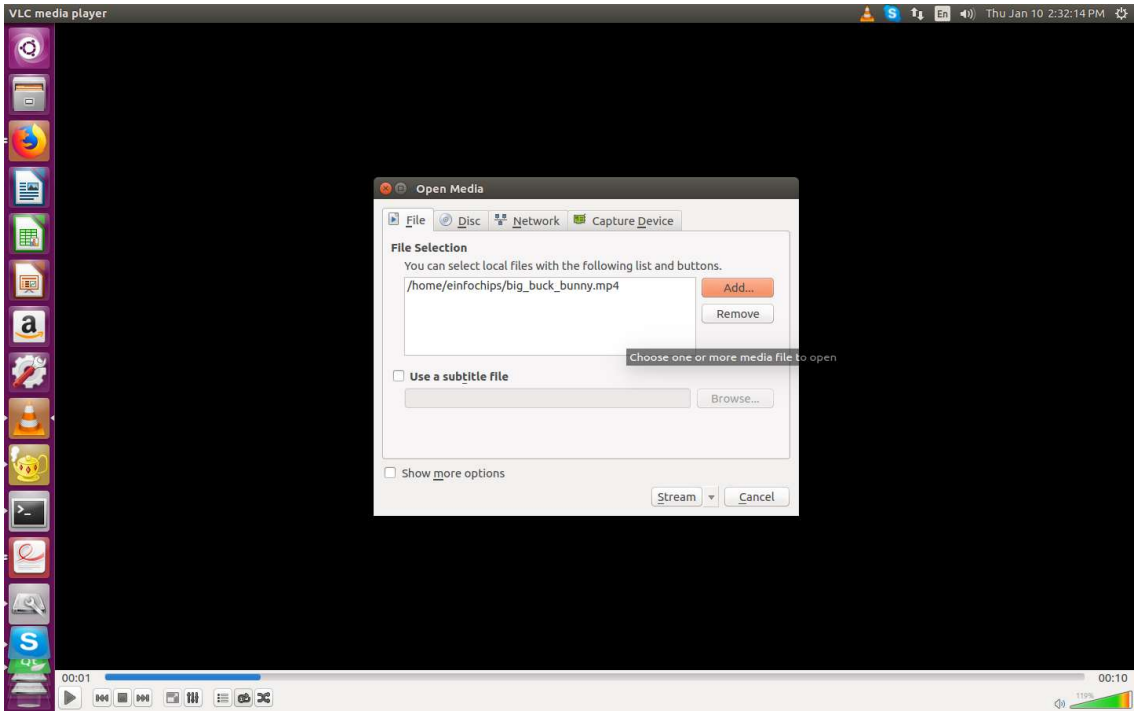

Figure 7 : Add Video file for RTSP stream

- Provide any video file that we want to stream. i.e. big_buck_bunny.mp4

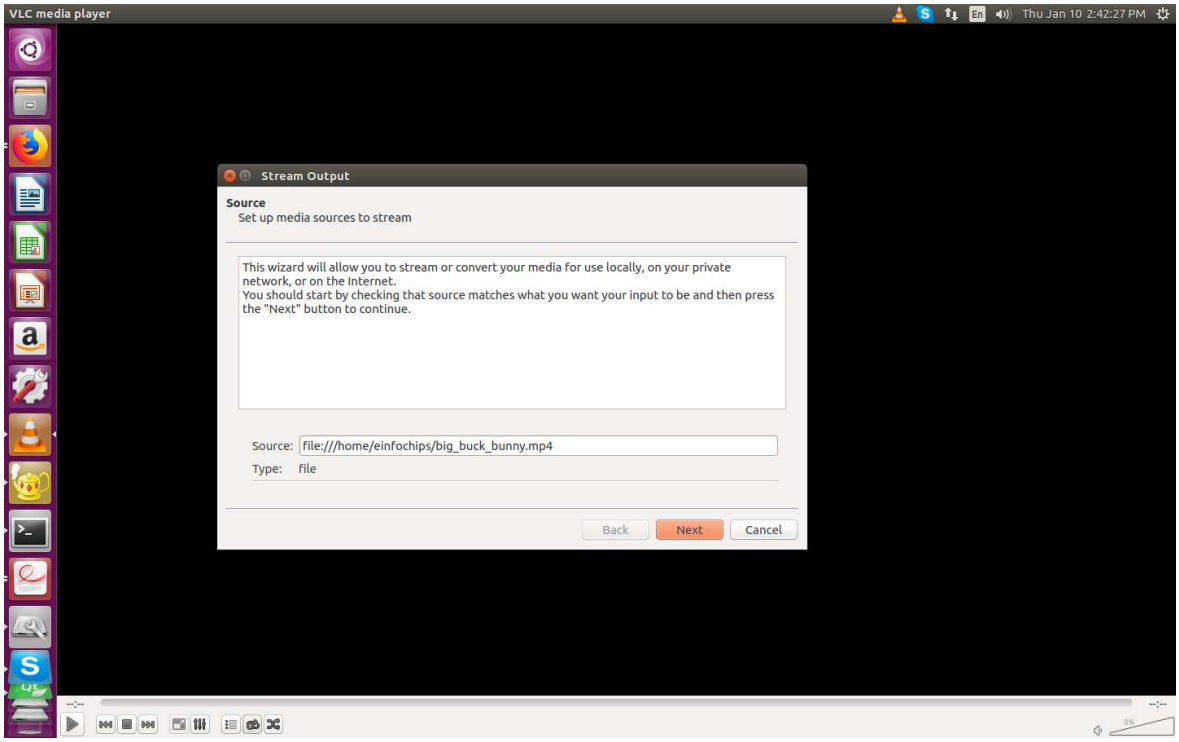

Figure 8 : Verify Video details

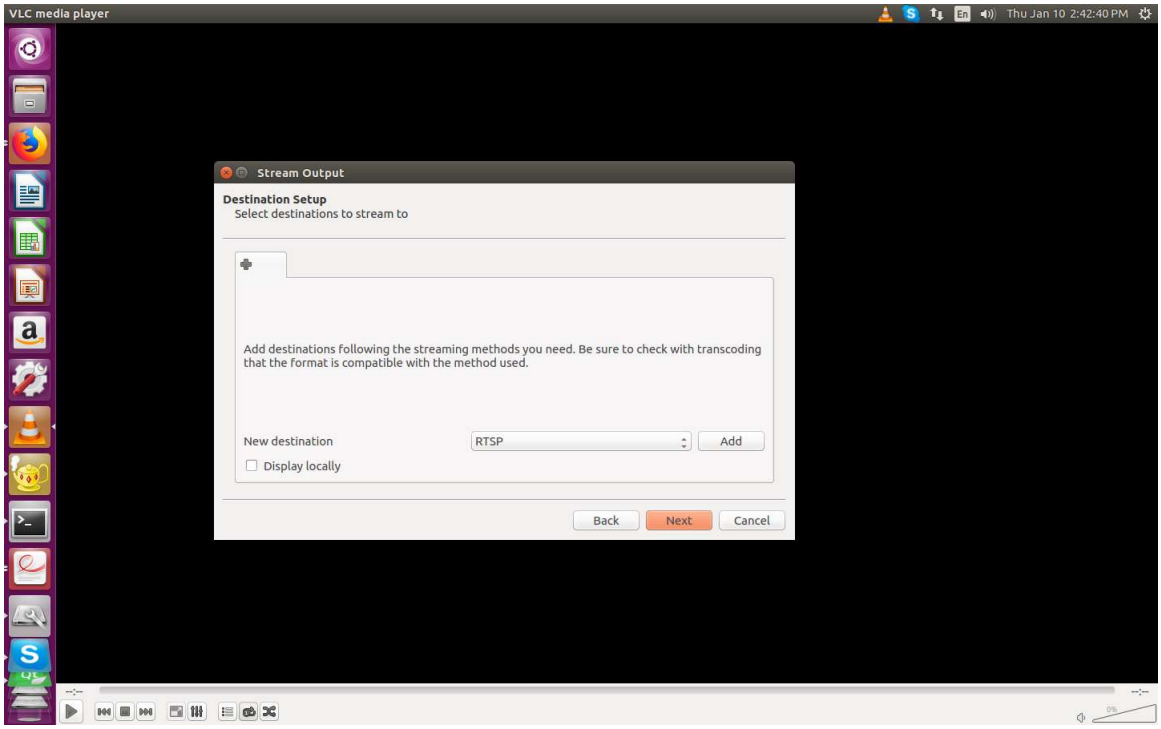

Figure 9 : Select RTSP option

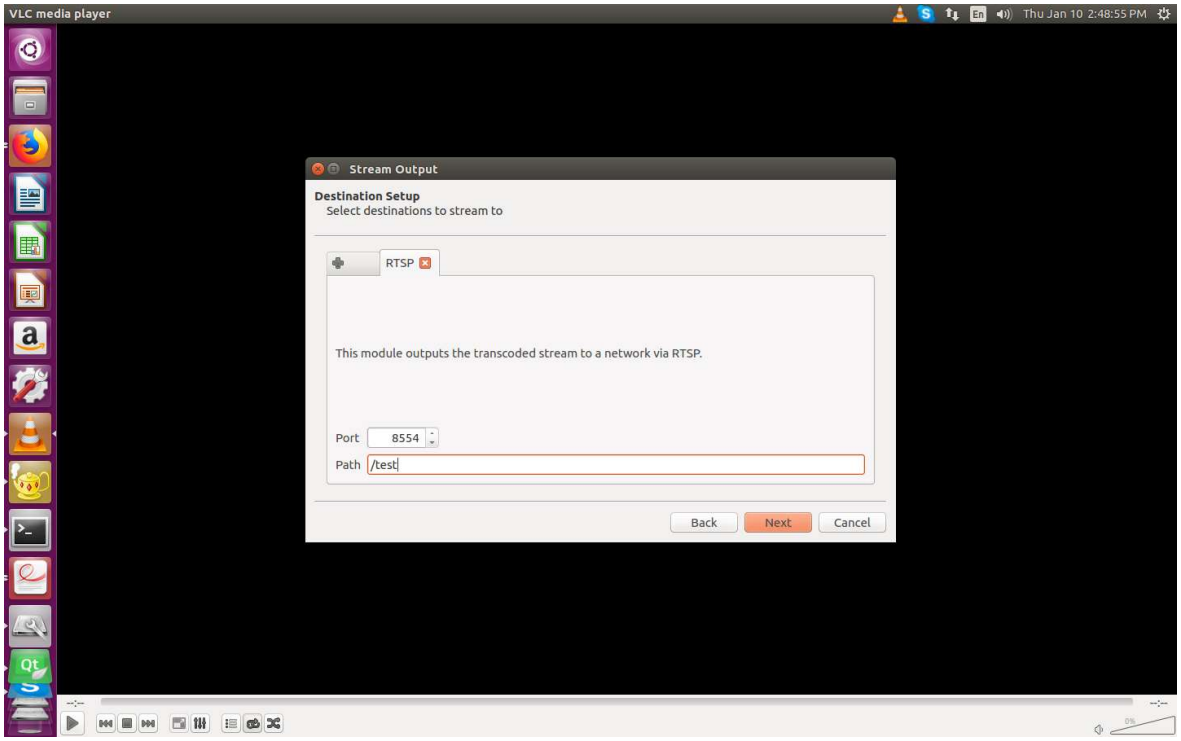

Figure 10 : Provide video stream name

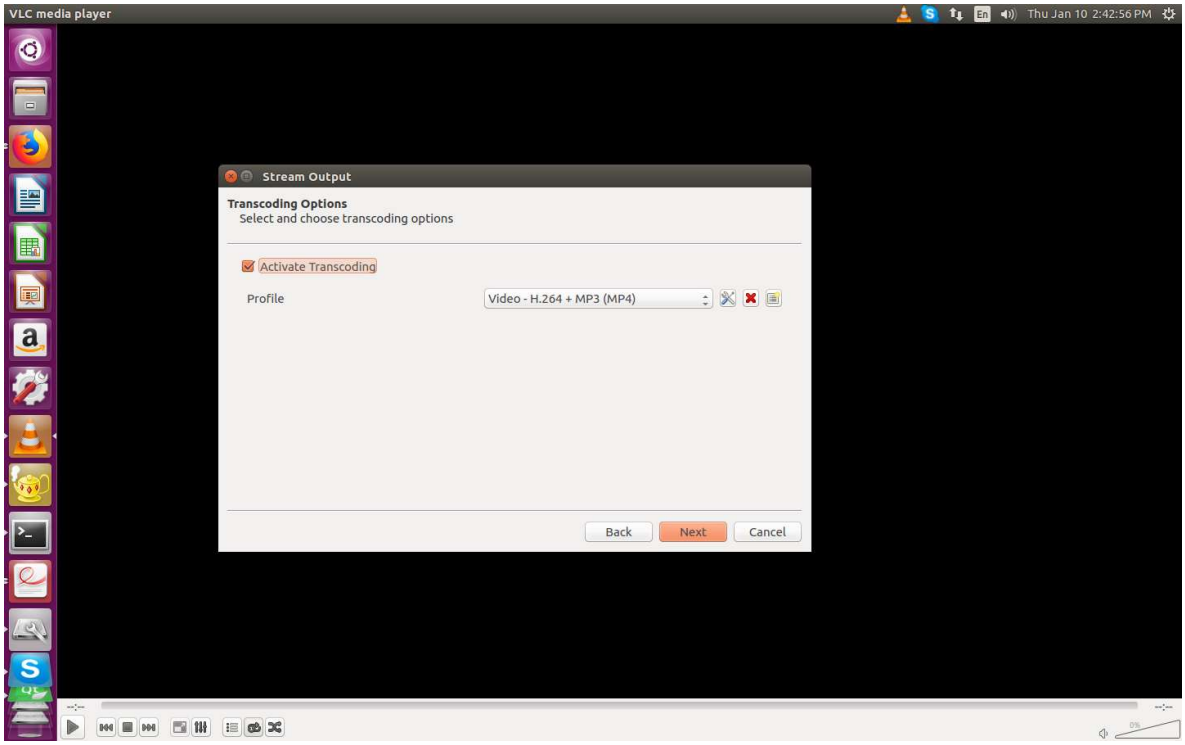

Figure 11 : Select Video codec

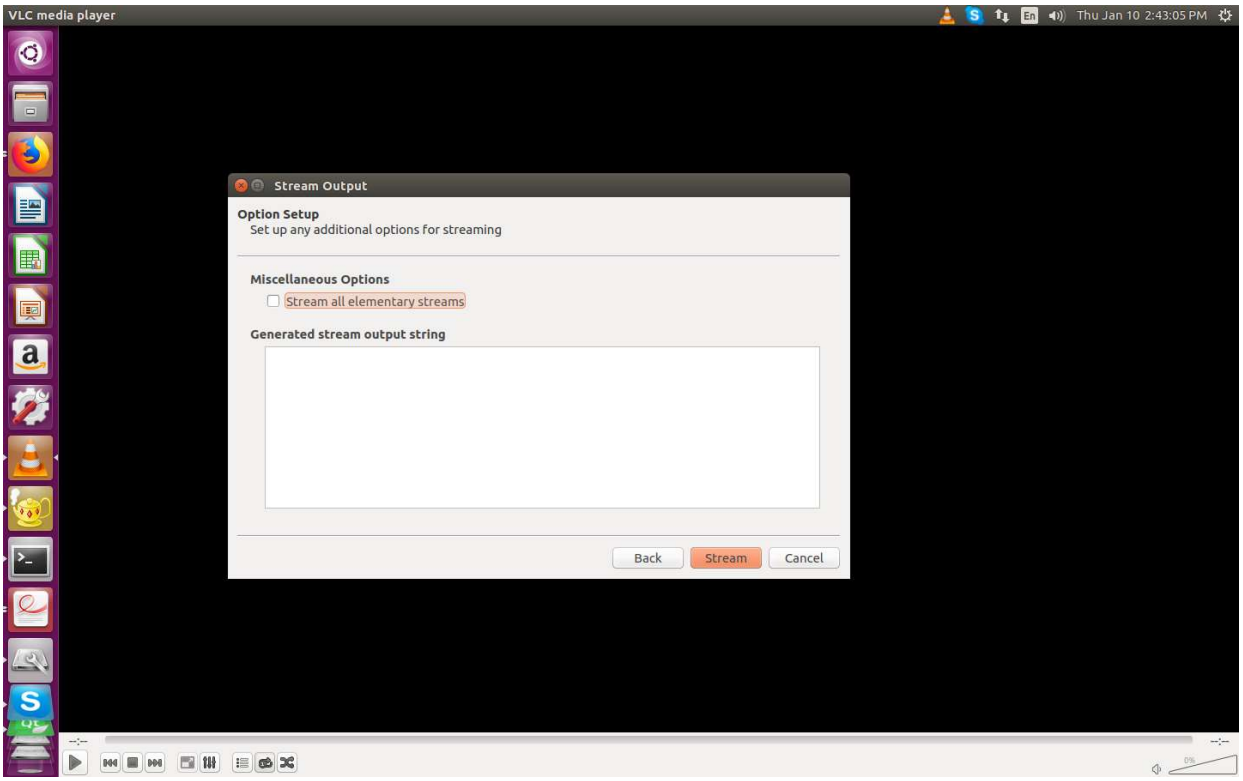

Figure 12 : Start Network Streaming

- After above steps VLC work as network stream and make sure that stream should be in loop. Therefore, after finishing video it starts over again.
- Now set static IP on board as well.
  # ifconfig eth0 192.168.1.2

Play Network stream and local videos on HDMI display (Dual)

- Ensure the network bandwidth.
- To play network stream on DSI (HDMI2 – j15) display, create an rtsp server on any host machine (i.e. x86) and play video over network, (Follow above steps) and note IP address and port number for rtsp server (host machine).
- In board's console apply below command to play the same stream over the network

  gst-launch-1.0 playbin uri=rtsp://<Ip_add>:<port>/<video stream name > uridecodebin0::source::latency=300 &

  E.g. based on above setting for VLC and local LAN:
  # gst-launch-1.0 rtspsrc location=rtsp://192.168.1.1:8554/test ! decodebin ! kmssink sync=true &

- Play simultaneously local video over HDMI1 (j2) using below command.
  # gst-launch-1.0 filesrc location=sample_video.mp4 typefind=true ! video/quicktime ! qtdemux ! queue max-size-time=0 ! vpudec ! queue max-size-time=0 ! waylandsink sync=true &

Play QT demo on HDMI display (Dual)

 # /usr/share/videohmiapp-1.0/VideoHMIApplication
 # /usr/share/chemicalplanthmi-1.0/ChemicalPlantHMI

Make sure that weston service should be started first.
For more details, see **Section 21 and 22** (QT Demos Details).

4. HDMI2 Touch Panel Demo

Play local videos on HDMI display

- Copy local sample_video.mp4 videos using scp command to on board sd card as mentioned in above demo.
- Insert HDMI cable into board's J15 HDMI2 connector and touch USB to board's USB.
- Apply power to board and go to terminal of x86 host system and open board's console as mentioned above
- Hold boot on u-boot screen by pressing any key on host machine keyboard (immediate after boot within 3 seconds)
- Apply dtb file changes as per below command on u-boot console

  # setenv fdt_file fsl-imx8mq-thor96-dcss-adv7535-b3.dtb

 # setenv mmcargs 'setenv bootargs ${jh_clk} console=${console} root=${mmcroot} video=HDMI-A-1:1280x800-12@70'
 # saveenv
 # boot

- Console will show booting logs
- Once booting is completed, console will hold on login prompt where user can enter username as **root**. (no password)
- Once boot completes apply below commands

  # systemctl stop weston
- Start playback using below command
  # gst-launch-1.0 filesrc location=video.mp4 typefind=true ! video/quicktime ! aiurdemux ! queue max-size-time=0 ! vpudec ! queue max-size-time=0 ! autovideosink

Touch Demo

- Ensure touch panel is connected to board using USB and is powered up.
- Go to board's console (require minicom) , log in and apply below commands

  # ls /dev/input/ -l
- Find entry as per below

  # lrwxrwxrwx 1 root 0 6 Nov 23 12:35 touchscreen0 -> event1
- Once received node, cat that node using below command,

  # cat /dev/input/touchscreen0 | hexdump
- Touch on the screen and get the events on console to validate touch

Play QT demo

 # /usr/share/videohmiapp-1.0/VideoHMIApplication
 # /usr/share/chemicalplanthmi-1.0/ChemicalPlantHMI

For more details, see **Section 21 and 22** (QT Demos Details).

5. Mezzanine DSI Display Demo

- Attach dsi display MX8_DSI_OLED to high-speed mezzanine connector.

Play video on mezzanine DSI display

- After verification on mezzanine connection with h/w, power up the board**.**
- Go to board's console (require minicom) and immediately stop at **u-boot autoboot** console by pressing any key.
- Apply below commands for changing dtb file

  # setenv fdt_file fsl-imx8mq-thor96-dcss-rm67191-b3.dtb
  # saveenv

 # boot

- Copy local sample_video.mp4 videos using scp command to on board sd card as mentioned in above demo.
- Play above copied video on the OLED Display using below command
  # gst-launch-1.0 filesrc location= sample_video.mp4 typefind=true ! video/quicktime ! aiurdemux ! queue max-size-time=0 ! vpudec ! queue max-size-time=0 ! autovideosink

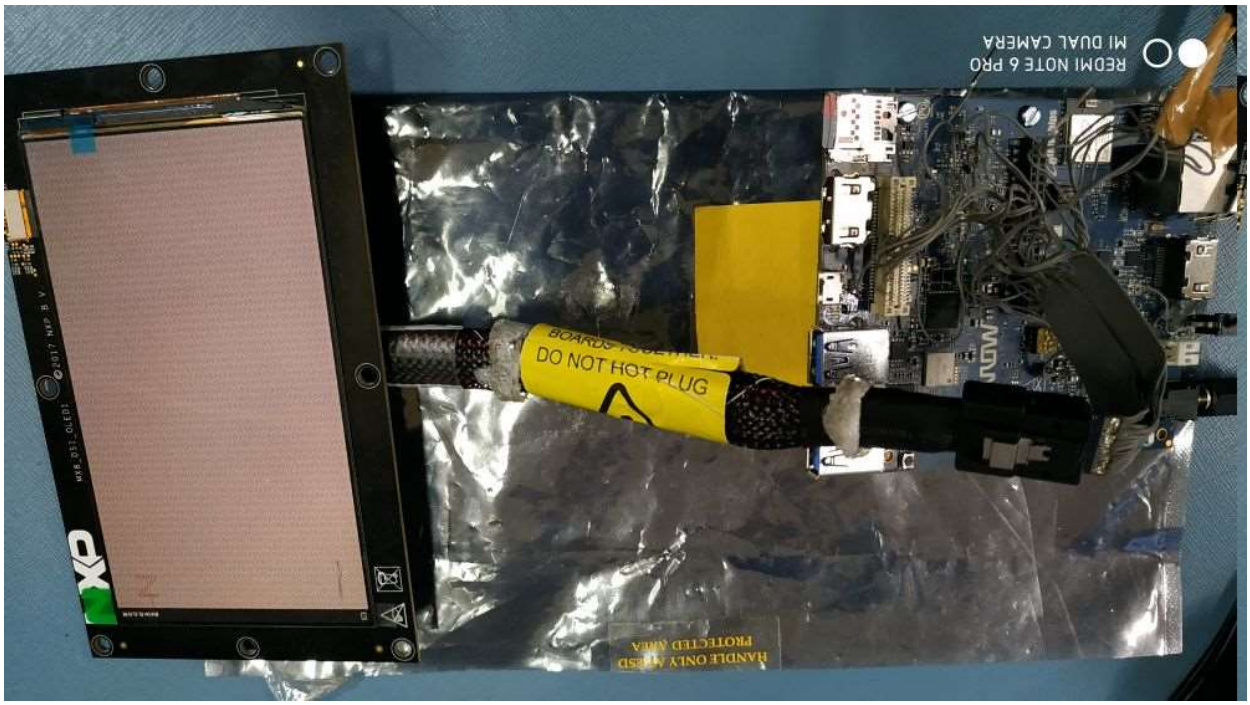

Figure 13 : iMX8M_Thor96 Platform Mezzanine DSI OLED

6. Camera Demo

Live stream from camera on HDMI display

- To watch live stream over the HDMI, connect HDMI Display.
- Power up the board
- Go to board's console (require minicom) and immediately stop at **u-boot autoboot** console by pressing any key.
- Apply below commands for changing dtb file

  # setenv fdt_file fsl-imx8mq-thor96-mipi-csi.dtb
  # saveenv
  # boot
- Display will get blank and will be black only
- Attach camera module to high-speed mezzanine on CSI2 connector.
- Then apply below command

  # gst-launch-1.0 v4l2src device=/dev/video1 ! video/x-raw,width=1280,height=720 ! kmssink

- Attach camera module to high-speed mezzanine on CSI1connector.
- Then apply below command

  # gst-launch-1.0 v4l2src device=/dev/video0 ! video/x-raw,width=1280,height=720 ! kmssink

- This will show live streaming over the attached HDMI for couple of minutes

Note: Dual camera mode is not tested yet. Please perform above steps on single camera mode.

Live stream from camera on HDMI1 display and local Video streaming on HDMI2(DSI-HDMI)

- To watch live stream over the HDMI, connect HDMI Displays to HDMI1(J2) and HDMI2(j15) connector.
- Attach camera module to high-speed mezzanine on CSI2 connector.
- Power up the board
- Go to board's console (require minicom) and immediately stop at **u-boot autoboot** console by pressing any key.
- Apply below commands for changing dtb file

  # setenv fdt_file fsl-imx8mq-thor96-dual-display-b3.dtb
  # saveenv
  # boot
- Command to camera image on HDMI1

  # gst-launch-1.0 v4l2src device=/dev/video1 ! video/x-raw,width=1280,height=720 ! kmssink

- Command to play video on HDMI2.

  # gst-launch-1.0 -v filesrc location=sample_video.mp4 typefind=true ! video/quicktime ! aiurdemux ! queue max-size-time=0 ! vpudec ! waylandsink &

- This will show live streaming over the attached HDMI for couple of minutes

Capture image from camera

- Go to board's console (require minicom) and power up the board with above mentioned dtb change configuration.
- Ensure Ethernet is plugged-in to get image from board to local x86 host pc.
- Apply below command to capture image from camera.

  # gst-launch-1.0 v4l2src num-buffers=1 ! jpegenc ! filesink location= /home/root/test.jpg

- Above command will capture image named test.jpg in /home/root/ location
  Copy image from board to local pc using below command

  # scp test.jpg <user name of host pc >@<ip of host pc>:/home/user/Desktop
- Go to local pc's /home/user/Desktop and watch image into image viewer to verify captured image from board's camera.

7. Audio Codec Demo

- Go to board's console (require picocom) and power up.
- Install picocom in host machine using below command.
  # sudo apt-get install picocom

- After installing picocom open console use of below command
  # sudo picocom -b 115200 -r -l /dev/ttyUSB0

- From the above command you will get the imx8mq thor96 board's tty console
- Type alsamixer command in console
- Following screen will get on console.

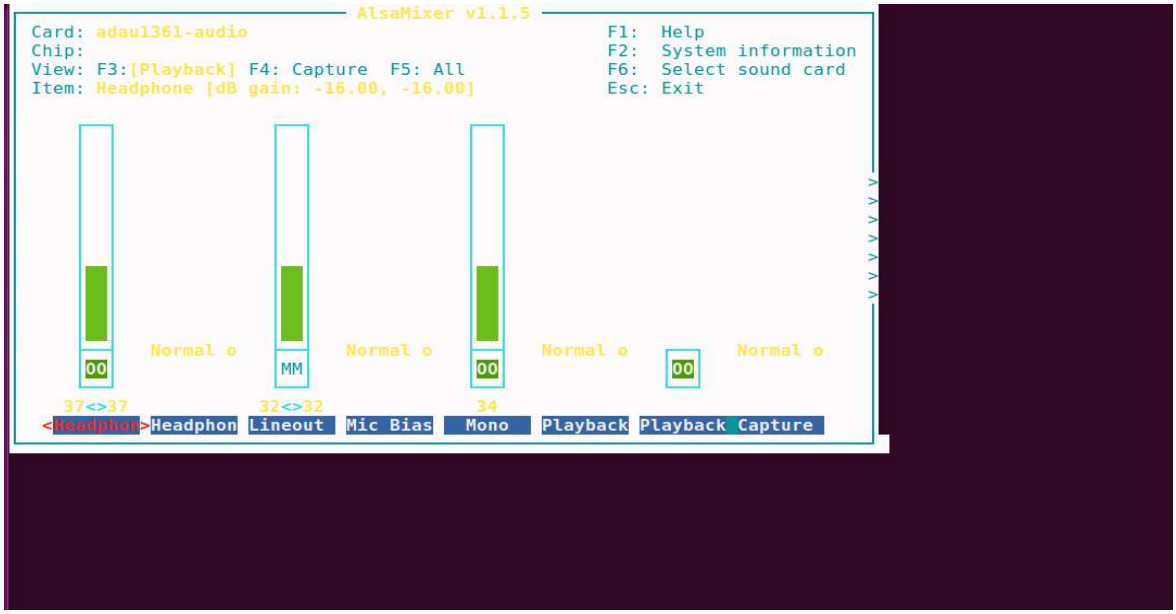

Figure 14 : Alsa Mixer Control Panel

- Note: Above **alsamixer** command's required settings are already set by default and user do not need to do anything to play or record audio. However, if user want to adjust volume or other setting then he/she can change settings based on his/her preferences.
- You can select a different sound card by pressing **F6**. It will bring up a menu that shows the known sound cards on imx8mq thor96 system.
- The default you see above is the "Playback" view. You can choose "Capture" by pressing F4 and "All" (which includes "Playback" and "Capture") by pressing **F5**. Return to "Playback" with F3.Move right and left, respectively, through those options by pressing the Left and Right arrow keys.
- Adjust each **volume with Down** to reduce the volume of a channel and **Up to increase** the volume.
- You can **mute** and **unmute** any channel by **pressing m.**

- To play and capture the audio you need to all setting must be **unmute** and gain of all channel are not to be **zero**.
- After done all setting press ESC button to close alsamixer utility.
- Enter below command to check the audio codec playback probed
  # aplay -l

- find the below log from the list
  adau1361audio [adau1361-audio], device 0: adau1x61 adau-hifi-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
- Enter below command to check audio codec capture driver probed
  # arecord -l

- Find below logs from the list
  adau1361audio [adau1361-audio], device 0: adau1x61 adau-hifi-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
- To playback audio enter below command
  # aplay -Dplughw:0,0 SAI1/sample.wav

- To record audio enter below command
  # arecord -Dplughw:0,0 -f dat record.wav

- To test playback with recording
  # arecord -Dplughw:0,0 -f dat | aplay -Dplughw:0,0 -f dat

8. LTE Demo

- Connect **Quectel** module with target board.
- Go To Board's console and apply below command
  # pppd call quectel-ppp &;

- Edit /etc/resolv.conf as per below
  # nameserver 59.144.127.117

  # nameserver 59.144.144.46

- Save above file and apply below command
  # ifconfig ppp0

  # ping www.google.com -I ppp0

- Will be able to ping to google.com

9. USB Hub demo

- Connect USB device disk to USB port of target board
- Go to board ‘s console and apply below command
  # lsusb

- On Connecting usb pendrive
  usb 2-1.2: new SuperSpeed USB device number 4 using xhci-hcd
  usb-storage 2-1.2:1.0: USB Mass Storage device detected
  scsi host0: usb-storage 2-1.2:1.0
  scsi 0:0:0:0: Direct-Access SanDisk Ultra Fit 1.00 PQ: 0 ANSI: 6
  sd 0:0:0:0: [sda] 30031872 512-byte logical blocks: (15.4 GB/14.3 GiB)

- On Disconnecting usb pendrive
  usb 2-1.2: USB disconnect, device number 4

10. USB OTG as host

- Power up the board
- Go to board's console (require minicom) and immediately stop at **u-boot autoboot** console by pressing any key.
- Apply below commands for changing dtb file

  # setenv fdt_file fsl-imx8mq-thor96-otg-host.dtb
  # saveenv
  # boot
- Once boot completes
- Connect USB device disk to USB OTG port of target board
- Go to board ‘s console and apply below command as same as USB hub
  lsusb
- On Connecting usb pendrive
  usb 2-1.2: new SuperSpeed USB device number 4 using xhci-hcd
  usb-storage 2-1.2:1.0: USB Mass Storage device detected
  scsi host0: usb-storage 2-1.2:1.0
  scsi 0:0:0:0: Direct-Access SanDisk Ultra Fit 1.00 PQ: 0 ANSI: 6
  sd 0:0:0:0: [sda] 30031872 512-byte logical blocks: (15.4 GB/14.3 GiB)

- On Disconnecting usb pendrive
  usb 2-1.2: USB disconnect, device number 4.

11. USB OTG as Devices

- Connect USB cable (same like debug uart cable) USB OTG port of target board
- Run the below command

```
# dd if=/dev/zero of=/mass_storage bs=1M seek=256 count=0
# mkfs.fat /mass_storage
# cat <<EOT | sfdisk --reorder  /mass_storage
 ,,c
 EOT
# mkfs.vfat /mass_storage
# chmod 777 /mass_storage
# mount -o loop  /mass_storage /mnt/
# mount
# modprobe  g_mass_storage file=/mass_storage
```

- Disconnect and connect the USB cable
- User will see the drive on host machine.

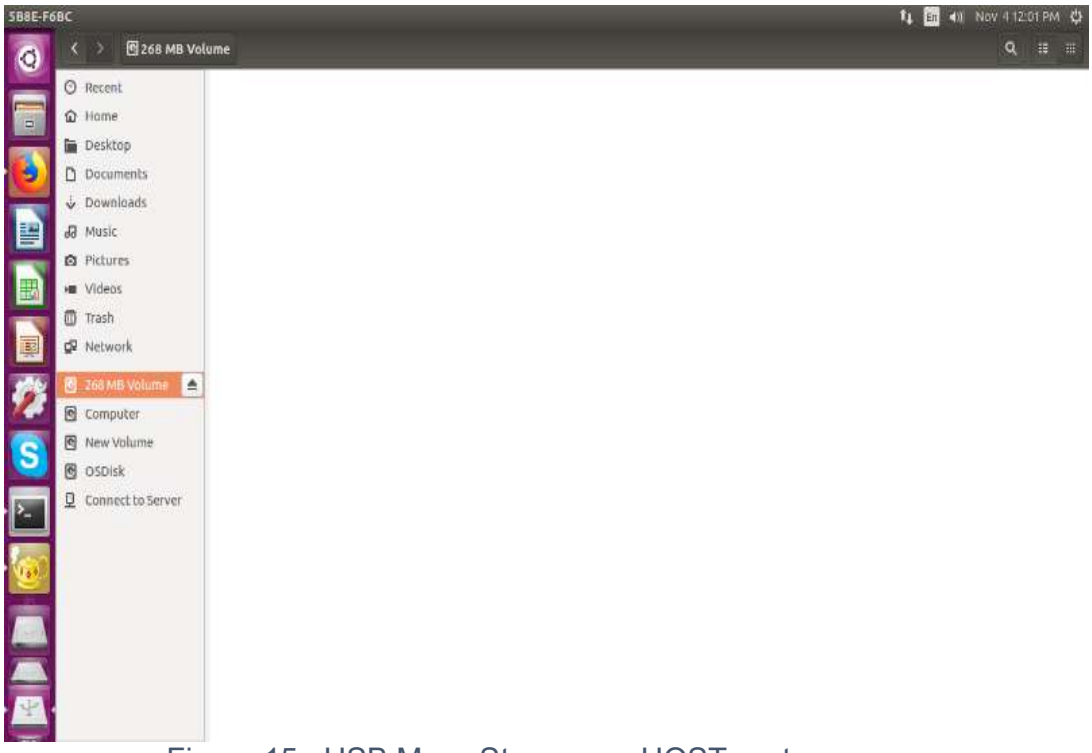

Figure 15 : USB Mass Storage on HOST system

Please note that on Window system mass storage been created but not seen the drive (although we have created FAT file system). Must be an issue with Windows system. Therefore, User need to test this with Linux system only.

12. Bluetooth

- Go to board ‘s console and apply below command
  # stty -F /dev/ttyUSB0 3000000
  # stty -F /dev/ttyUSB0 crtscts
  # hciattach /dev/ttyUSB0 bcm43xx 3000000 flow -t 20

- Wait until the complete the command response
  # hciconfig hci0 up
  # hciconfig hci0 -a

- User will get the hcio interface
- Run the “bluetoothctl” utility
  #bluetoothctl
  [bluetooth]# power on
  [bluetooth]# agent on
  [bluetooth]# default-agent
  [bluetooth]# pairable on
  [bluetooth]# scan on

  Copy mac address

  [bluetooth]# scan off
  [bluetooth]# pair <mac address>

  Approve pairing on Device if required

  [bluetooth]# trust <mac address>
  [bluetooth]# connect <mac address>
  [bluetooth]# quit

- Sending file command.
  #export $(dbus-launch)
  #/usr/libexec/bluetooth/obexd &

  #obexctl
  [obex]# connect <mac addr>
  [<mac addr>]# send <file>

  [<mac addr>]# disconnect
  [<mac addr>]# quit

- Play the audio over BT commands
- Collect the audio file from the support package folder.
- Get the Bluetooth headset or Bluetooth speaker.
  # aplay -D bluealsa:HCI=hci0,DEV=<mac addr>,PROFILE=a2dp play_audio.wav

- Get the Mobile headset,
- Connect mobile with our modem using above **bluetoothctl** command.
- play the music on mobile player
- run below command to capture the audio from Bluetooth
  # arecord -D bluealsa:HCI=hci0,DEV=<mac addr>,PROFILE=a2dp record_audio.wav
- copy recovered file in your host PC and verify with any player on host PC

13. EEPROM

- Run below command to test EEPROM
  # echo hello > /sys/bus/i2c/devices/i2c-1/1-0050/eeprom
  # cat /sys/bus/i2c/devices/i2c-1/1-0050/eeprom | hexdump -C

14. Zigbee Demo

Description:

ZigBee 3.0 Gateway:

ZigBee 3.0 provides a foundation of commissioning and network management mechanisms to be used in all ZigBee applications. The sample scenario presented here demonstrates the flexibility that the ZigBee 3.0 specification provides to applications. They also act as an excellent starting point for users wishing to build their own ZigBee 3.0 applications.

Z3Gateway, the gateway can form a centralized network, and the light and the switch can join the centralized network by performing network steering.

The gateway provides CLI commands (application interface) to the user. User can create new zigbee network and can remove using such CLI command set.

The CLI command "**plugin network-creator start 1**" create a centralized network. The gateway application can then be triggered to allow other zigbee devices connect to this network with the CLI command "**plugin network-creator-security open-network**". With this command we can create open-network where any reset zigbee device can join the network using the ZigBeeAlliance09 link key, or by manually entering the install code derived link key into the gateway using the CLI command "**plugin network-creator-security set-joining-link-key**". The CLI command "**plugin network-creator-security close-network**" will close the network and no longer allow devices onto the gateway's network.

Steps to test Zigbee as below:

- Copy host zigbee application to device using scp command via Ethernet if it is not there on the board at path **/home/root/zigbee/Z3GatewayHost** .
- Use below command to set executable flags for binary.
  # chmod 777 Z3GatewayHost

- Make sure “zigbee” service is disable. User can check status of zigbee service by “**systemctl status zigbee**”
- If Zigbee service is active running then disable it by “**systemctl stop zigbee**” and rename “**/home/root/zigbee/Z3GatewayHost_HMI**” to any other name.
- Run Z3GatewayHost as below. (Please flash ncp-spi and bootloader if not flashed). Note for board, which are shipped are already flashed.

  # ./Z3GatewayHost

  [../../../platform/base/hal/micro/unix/host/spi-protocol-linux.c] traceMask = 0xFF

  [../../../platform/base/hal/micro/unix/host/spi-protocol-linux.c] Opened nHOST_INT device 75.

  [../../../platform/base/hal/micro/unix/host/spi-protocol-linux.c] Opened SPI device /dev/spidev1.0.

  [../../../platform/base/hal/micro/unix/host/spi-protocol-linux.c] Opened nCS device 8.

  [../../../platform/base/hal/micro/unix/host/spi-protocol-linux.c] Opened nRESET device 132.

  [../../../platform/base/hal/micro/unix/host/spi-protocol-linux.c] Opened nWAKE device 74.

  [../../../platform/base/hal/micro/unix/host/spi-protocol-linux.c] Cannot write to /sys/class/gpio/export.

  [../../../platform/base/hal/micro/unix/host/spi-protocol-linux.c] Cannot write to /sys/class/gpio/export.

  [../../../platform/base/hal/micro/unix/host/spi-protocol-linux.c] Cannot write to /sys/class/gpio/export.

  [../../../platform/base/hal/micro/unix/host/spi-protocol-linux.c] Cannot write to /sys/class/gpio/export.

  [../../../platform/base/hal/micro/unix/host/spi-protocol-linux.c] traceMask = 0xFF

  [../../../platform/base/hal/micro/unix/host/spi-protocol-linux.c] Opened nHOST_INT device 75.

  [../../../platform/base/hal/micro/unix/host/spi-protocol-linux.c] Opened SPI device /dev/spidev1.0.

  [../../../platform/base/hal/micro/unix/host/spi-protocol-linux.c] Opened nCS device 8.

  [../../../platform/base/hal/micro/unix/host/spi-protocol-linux.c] Opened nRESET device 132.

  [../../../platform/base/hal/micro/unix/host/spi-protocol-linux.c] Opened nWAKE device 74.

  Reset info: 11 (SOFTWARE)

  ezsp ver 0x07 stack type 0x02 stack ver. [6.4.0 GA build 385]

Ezsp Config: set source route table size to 0x0064:Success: set

Ezsp Config: set security level to 0x0005:Success: set

Ezsp Config: set address table size to 0x0002:Success: set

Ezsp Config: set TC addr cache to 0x0002:Success: set

Ezsp Config: set stack profile to 0x0002:Success: set

Ezsp Config: set MAC indirect TX timeout to 0x1E00:Success: set

Ezsp Config: set max hops to 0x001E:Success: set

Ezsp Config: set tx power mode to 0x8000:Success: set

Ezsp Config: set supported networks to 0x0001:Success: set

Ezsp Policy: set binding modify to "allow for valid endpoints & clusters only":Success: set

Ezsp Policy: set message content in msgSent to "return":Success: set

Ezsp Value : set maximum incoming transfer size to 0x00000052:Success: set

Ezsp Value : set maximum outgoing transfer size to 0x00000052:Success: set

Ezsp Config: set binding table size to 0x0010:Success: set

Ezsp Config: set key table size to 0x0000:Success: set

Ezsp Config: set max end device children to 0x0020:Success: set

Ezsp Config: set aps unicast message count to 0x000A:Success: set

Ezsp Config: set broadcast table size to 0x000F:Success: set

Ezsp Config: set neighbor table size to 0x0010:Success: set

NCP supports maxing out packet buffers

Ezsp Config: set packet buffers to 255

Ezsp Config: set end device poll timeout to 0x0005:Success: set

Ezsp Config: set end device poll timeout shift to 0x0006:Success: set

Ezsp Config: set zll group addresses to 0x0000:Success: set

Ezsp Config: set zll rssi threshold to 0xFF80:Success: set

Ezsp Config: set transient key timeout to 0x00B4:Success: set

Ezsp Endpoint 1 added, profile 0x0104, in clusters: 8, out clusters 19

Ezsp Endpoint 242 added, profile 0xA1E0, in clusters: 0, out clusters 1

Found 0 files

Z3GatewayHost>**network leave**

leave 0x70

Z3GatewayHost> **plugin network-creator start 1**

NWK Creator Security: Open network: 0x01

Z3GatewayHost> **plugin network-creator-security open-network**

NWK Creator: Form: 0x00

NWK Creator Security: Start: 0x00

NWK Creator: Form. Channel: 20. Status: 0x00

NWK Creator: Stop. Status: 0x00. State: 0x00

EMBER_NETWORK_UP 0x0000

Now turn on a zigbee end device and it will be start broadcasting, will connect above network, and on host side below log will occur.

Z3GatewayHost>Trust Center Join Handler: status = UNsecured join, decision = use preconfigured key (00), shortid 0x2229

T0000007E:RX len 3, ep FF, clus 0x0003 (Identify) FC 01 seq 00 cmd 01 payload[]

T0000007E:RX len 4, ep 01, clus 0x0500 (IAS Zone) FC 08 seq 00 cmd 04 payload[00]

T0000007F:RX len 15, ep 01, clus 0x0500 (IAS Zone) FC 08 seq 01 cmd 01 payload[10 00 00 F0 16 BC 1E FE FF 9F FD 90]

T0000007F:RX len 12, ep 01, clus 0x0019 (Over the Air Bootloading) FC 01 seq 01 cmd 01 payload[00 31 11 24 10 21 51 00 23]

QueryNextImageRequest mfgId:0x1131 imageTypeId:0x1024, fw:0x23005121

T0000007F:RX len 20, ep 01, clus 0x0500 (IAS Zone) FC 08 seq 02 cmd 01 payload[00 00 00 30 00 01 00 00 31 15 00 02 00 00 19 20 00]

T0000008F:RX len 7, ep 01, clus 0x0500 (IAS Zone) FC 09 seq 02 cmd 01 payload[15 00 31 11]

Sent enroll response with responseCode: 0x00, zoneId: 0x00, status: 0x00

Below log is for door sensor device which indicates open<**24**> and close<**25**>:

T00000092:RX len 9, ep 01, clus 0x0500 (IAS Zone) FC 09 seq 03 cmd 00 payload**[24** 00 00 00 00 00]

T0000009A:RX len 9, ep 01, clus 0x0500 (IAS Zone) FC 09 seq 04 cmd 00 payload**[25** 00 00 00 00 00]

T000000A0:RX len 9, ep 01, clus 0x0500 (IAS Zone) FC 09 seq 05 cmd 00 payload[24 00 00 00 00 00]

T000000A2:RX len 9, ep 01, clus 0x0500 (IAS Zone) FC 09 seq 06 cmd 00 payload[25 00 00 00 00 00]

T000000A5:RX len 9, ep 01, clus 0x0500 (IAS Zone) FC 09 seq 07 cmd 00 payload[24 00 00 00 00 00]

T000000A7:RX len 9, ep 01, clus 0x0500 (IAS Zone) FC 09 seq 08 cmd 00 payload[25 00 00 00 00 00]

T000000A8:RX len 9, ep 01, clus 0x0500 (IAS Zone) FC 09 seq 09 cmd 00 payload[24 00 00 00 00 00]

T000000A9:RX len 9, ep 01, clus 0x0500 (IAS Zone) FC 09 seq 0A cmd 00 payload[25 00 00 00 00 00]

T000000B2:RX len 9, ep 01, clus 0x0500 (IAS Zone) FC 09 seq 0B cmd 00 payload[21 00 00 00 00 00]

T000000B4:RX len 9, ep 01, clus 0x0500 (IAS Zone) FC 09 seq 0C cmd 00 payload[25 00 00 00 00 00]

T000000C4:RX len 3, ep FF, clus 0x0003 (Identify) FC 01 seq 0D cmd 01 payload[]

T000000C8:RX len 3, ep FF, clus 0x0003 (Identify) FC 01 seq 0E cmd 01 payload[]

15. Thread Demo

Description:

Client/Server Sample Applications

The client and server applications demonstrate basic Thread network functionality for a wireless sensor network. This server application acts as a data sink and collects information from client nodes that act as sensors. The client and server communicate using the Constrained Application Protocol (CoAP) at the application layer, with UDP serving as the transport layer. The Silicon Labs Thread stack provides CoAP and UDP.

At startup, the server will automatically start network operations. If the node is starting for the first time, it will form a new network called "client/server." If it had already formed a network previously, it will simply resume network operations using the network parameters stored in non-volatile memory in the stack. After forming or resuming, the application establishes itself as the commissioning device of the network. This means the server is responsible for allowing other devices to join the network.

The server sends advertisement messages to the network at regular intervals using multicast transmission. The advertisements, sent as CoAP POST requests to the "server/advertise" URI, inform clients in the network about the presence of the server. When a client without a server receives the advertisement, it will begin sending sensor data to the server. Sensor data are sent as CoAP POST requests to the "client/report" URI, and are unicast directly to the server. In these examples, the sensor data are the current temperature, as provided by the temperature sensor on the development board of the client nodes.

Before attempting to join the network, the client nodes print their unique join key to the console. This key must be provided to the commissioner (i.e., the server) before the client will be able to join the network. For example, when joining, the client will display a message such as:

Joining network "client/server" with EUI64 >0134047823560034 and join key "174F5B07"

In this example, "174F5B07" is the unique join key for the client and 0134047823560034 is the EUI64 of the client. The join key can be provided to the server and joining can be enabled via the following CLI command:

expect "174F5B07"

If the EUI64 of the client is known, it can be specified in order to further assist in joining the correct node:

expect "174F5B07" \\{0134047823560034\\}

It is important to note that each client will have a different join key. It is essential that the server be informed of the exact join key used by the client. If the join key is not provided to the server, or if an incorrect key is provided, the client will not be able to join. The server host sample application can communicate with the NCP using either SPI or UART. With SPI, the spi-server and ip-driver-app utility programs are used to interact with the NCP.

With UART, only the ip-driver-app utility program is used. spi-server will run itself in the background. ip-driver-app runs in the foreground by default, but it can be run in the background using shell job control features.

Steps to test thread as below:

Server Host appliation.

- move to directory path /home/root/thread/imx_host_apps/
  # cd thread/imx_host_apps/
  # cd spi-server
- Run spi_server.sh script with required parameter.
  # ./spi-server.sh 4951 spidev1.0 75 132 74 0xFF 8 --nolog &
  # cd ../
- Run **ip-driver-app**
  # ./ip-driver-app -s -u 4951 -t tun0 -m 4901 &

Run host server application as below.(Please flash ncp-spi for thread and bootloader if not flashed). For **join key** which is unique for each client we need to run client application and get it from their. Please see Client log in next section.

 # ./server-host -m 4901

```
Reset info: 0x0B (SOFTWARE)
Removing any IPv6 addresses configured on the host...
Init: 0x00
Resuming operation on network "client/server"
[2018-12-05 13:41:19.036 +   0.002] [app->driver->NCP ] [MGMT] [016907]
                                [NCP->driver mgmt ] [MGMT]
[016302075468726561640000F00280001810315446563203230203230313820313631363A35
303A353800]
[2018-12-05 13:41:19.037 +   0.001] [driver->app ] [MGMT]
[016302075468726561640000F00280001810315446563203230203230313820313631363A35
303A353800]
Host: Thread 2.7.1.0 GA build 245 management 3840 (Nov  6 2018 14:38:44)
NCP:  Thread 2.8.0.0 GA build 385 management 3840 (Dec 20 2018 16:50:58)
[2018-12-05 13:41:19.080 +   0.043] [NCP->driver mgmt ] [MGMT]
[01637A10636C69656E742F736572766572000000008FDE185ACB16E000008E185ACB
16E6C0AE8DBBC1A020305081EAD1EFEFF9FFD90085B5]
                                [driver->app ] [MGMT]
[01637A10636C69656E742F736572766572000000008FDE185ACB16E000008E185ACB
16E6C0AE8DBBC1A020305081EAD1EFEFF9FFD90085B509BE6]
[2018-12-05 13:41:19.081 +   0.001] [NCP->driver mgmt ] [MGMT]
[01637208E185ACB16E6C0AE810636C69656E742F736572766572000000008FDE185A
CB16E0000DBBC1A081EAD1EFEFF9FFD90085B509BE6A]
                                [driver->app ] [MGMT]
[01637208E185ACB16E6C0AE810636C69656E742F736572766572000000008FDE185A
CB16E0000DBBC1A081EAD1EFEFF9FFD90085B509BE6A7EC18]
[2018-12-05 13:41:19.082 +   0.001] [NCP->driver mgmt ] [MGMT] [016375050100]
                                [driver->app ] [MGMT] [016375050100]
                                [NCP->driver mgmt ] [MGMT] [01630C00]
```

[driver->app] [MGMT] [01630C00]
[NCP->driver mgmt] [MGMT]
[01637A10636C69656E742F7365727665720000008FDE185ACB16E000008E185ACB
16E6C0AE8DBBC1A020305081EAD1EFEFF9FFD90085B5]
[driver->app] [MGMT]
[01637A10636C69656E742F7365727665720000008FDE185ACB16E000008E185ACB
16E6C0AE8DBBC1A020305081EAD1EFEFF9FFD90085B509BE6]
[2018-12-05 13:41:19.083 + 0.001] [NCP->driver mgmt] [MGMT]
[01637208E185ACB16E6C0AE810636C69656E742F7365727665720000008FDE185A
CB16E0000DBBC1A081EAD1EFEFF9FFD90085B509BE6A]
[driver->app] [MGMT]
[01637208E185ACB16E6C0AE810636C69656E742F7365727665720000008FDE185A
CB16E0000DBBC1A081EAD1EFEFF9FFD90085B509BE6A7EC18]

[2018-12-05 13:41:20.563 + 1.480] [NCP->driver mgmt] [MGMT]
[01637A10636C69656E742F7365727665720000008FDE185ACB16E000008E185ACB
16E6C0AE8DBBC1A020305081EAD1EFEFF9FFD90085B5]
[2018-12-05 13:41:20.564 + 0.001] [driver->app] [MGMT]
[01637A10636C69656E742F7365727665720000008FDE185ACB16E000008E185ACB
16E6C0AE8DBBC1A020305081EAD1EFEFF9FFD90085B509BE6]
[2018-12-05 13:41:20.567 + 0.003] [NCP->driver mgmt] [MGMT]
[01637208E185ACB16E6C0AE810636C69656E742F7365727665720000008FDE185A
CB16E0000DBBC1A081EAD1EFEFF9FFD90085B509BE6A]
[2018-12-05 13:41:20.568 + 0.001] [driver->app] [MGMT]
[01637208E185ACB16E6C0AE810636C69656E742F7365727665720000008FDE185A
CB16E0000DBBC1A081EAD1EFEFF9FFD90085B509BE6A7EC18]
[NCP->driver mgmt] [MGMT] [016375050500]
[driver->app] [MGMT] [016375050500]

Bound to fde1:85ac:b16e:0:2f36:83f4:774:df19
Bound to fe80::ad18:eca7:e69b:505b
Resumed operation on network "client/server"
Becoming commissioner "server"
Became commissioner
Using the following thread multicast addresses:
ff32:40:fde1:85ac:b16e::1
ff33:40:fde1:85ac:b16e::1

Advertising to ff33:40:fde1:85ac:b16e::1
[2018-12-05 13:43:36.660 + 60.184] [IP stack->driver->NCP] [DATA]
[60001641001F110AFDE185ACB16E00002F3683F40774DF19FF330040FDE185ACB1
6E000000000001163316330 01F542F5202544CB8]
[2018-12-05 13:43:36.668 + 0.008] [NCP->driver->IP stack] [DATA]
[60001641001F110AFDE185ACB16E00002F3683F40774DF19FF330040FDE185ACB1
6E000000000001163316330 01F542F5202544CB8]

server-host> help
Usage notes:
type description
<uint8_t> 8-bit unsigned int, eg: 255, 0xAB
<int8_t> 8-bit signed int, eg: -128, 0xA9

<uint16_t> 16-bit unsigned int, eg: 3000 0xFFAA
<string> A string, eg: "foo" or {0A 1B 2C}
* Zero or more of the previous type
advertise
bootloader...
coap...
coaps...
exit
expect <string> <string> *
help
icmp...
info
network-management...
reset
udp...
versions
server-host> expect "GHHFBM02"

Sent steering data

[2018-12-05 13:48:37.980 + 0.167] [IP stack->driver->NCP] [DATA]
[600A612D000C1140FDE185ACB16E00002F3683F40774DF19FDE185ACB16E000074
34C3C136E3028C16331633000C02D66000FB81]
Received 43000 from client at fde1:85ac:b16e:0:7434:c3c1:36e3:28c
[2018-12-05 13:48:38.081 + 0.101] [IP stack->driver->NCP] [DATA]
[600A612D000E1140FDE185ACB16E00002F3683F40774DF19FDE185ACB16E000074
34C3C136E3028C16331633000E197E6244FB83E7]

[2018-12-05 13:48:48.227 + 10.146] [NCP->driver->IP stack] [DATA]
[6000000000211140FDE185ACB16E00007434C3C136E3028CFDE185ACB16E00002F
3683F40774DF19163316330021F0634202FB84E7]
Received 43000 from client at fde1:85ac:b16e:0:7434:c3c1:36e3:28c
[2018-12-05 13:48:48.260 + 0.033] [IP stack->driver->NCP] [DATA]
[600A612D000E1140FDE185ACB16E00002F3683F40774DF19FDE185ACB16E000074
34C3C136E3028C16331633000E197C6244FB84E7]

[2018-12-05 13:48:58.406 + 10.146] [NCP->driver->IP stack] [DATA]
[6000000000211140FDE185ACB16E00007434C3C136E3028CFDE185ACB16E00002F
3683F40774DF19163316330021F0614202FB85E7]
Received 43000 from client at fde1:85ac:b16e:0:7434:c3c1:36e3:28c
[2018-12-05 13:48:58.440 + 0.034] [IP stack->driver->NCP] [DATA]
[600A612D000E1140FDE185ACB16E00002F3683F40774DF19FDE185ACB16E000074
34C3C136E3028C16331633000E197A6244FB85E7]

[2018-12-05 13:49:08.585 + 10.145] [NCP->driver->IP stack] [DATA]
[6000000000211140FDE185ACB16E00007434C3C136E3028CFDE185ACB16E00002F
3683F40774DF19163316330021F05F4202FB86E7]
Received 43000 from client at fde1:85ac:b16e:0:7434:c3c1:36e3:28c
[2018-12-05 13:49:08.619 + 0.034] [IP stack->driver->NCP] [DATA]
[600A612D000E1140FDE185ACB16E00002F3683F40774DF19FDE185ACB16E000074
34C3C136E3028C16331633000E19786244FB86E7]

[2018-12-05 13:49:18.765 + 10.146] [NCP->driver->IP stack] [DATA]
[6000000000211140FDE185ACB16E00007434C3C136E3028CFDE185ACB16E00002F
3683F40774DF19163316330021F05B4202FB88E7]
Received 43000 from client at fde1:85ac:b16e:0:7434:c3c1:36e3:28c
[2018-12-05 13:49:18.799 + 0.034] [IP stack->driver->NCP] [DATA]
[600A612D000E1140FDE185ACB16E00002F3683F40774DF19FDE185ACB16E000074
34C3C136E3028C16331633000E19746244FB88E7]

[2018-12-05 13:49:28.945 + 10.146] [NCP->driver->IP stack] [DATA]
[6000000000211140FDE185ACB16E00007434C3C136E3028CFDE185ACB16E00002F
3683F40774DF19163316330021F0594202FB89E7]
Received 43000 from client at fde1:85ac:b16e:0:7434:c3c1:36e3:28c

Client Host application:

For Client application, please use a ThunderBoard and flash images from ThunderBoard_Bootloader_client. When board is up and will be able to see console.
client> join
client> Joining network "client/server" with EUI64 >90FD9FFFFE5FD1DD and join key "GHHFBM02"
ERR: Joining failed: 0x02

client> Joined network "client/server"
Waiting for an advertisement from a server
Attached to server at fde1:85ac:b16e:0:2f36:83f4:774:df19
Reporting 43000 to server at fde1:85ac:b16e:0:2f36:83f4:774:df19
Reporting 43000 to server at fde1:85ac:b16e:0:2f36:83f4:774:df19

When server uses join key "GHHFBM02**" and excepts it as it's client then client will send some data acting as a sensor and Server will receive logs for client sending data and server receiving is as below:**
Client :
Reporting 43000 to server at fde1:85ac:b16e:0:2f36:83f4:774:df19'
Server:
Received 43000 from client at fde1:85ac:b16e:0:7434:c3c1:36e3:28c

Note: Please refer Flash Document (FlashDocument.pdf) for the steps to flash.

16. USER LED

- Run the below command to control the Led

  - BT_LED

 # echo 96 > /sys/class/gpio/export
 # echo out > /sys/class/gpio/gpio96/direction
 # cat /sys/class/gpio/gpio96/value
 # echo 1 > /sys/class/gpio/gpio96/value

- WIFI_LED

 # echo 97 > /sys/class/gpio/export
 # echo out > /sys/class/gpio/gpio97/direction
 # cat /sys/class/gpio/gpio97/value
 # echo 1 > /sys/class/gpio/gpio97/value

- LED_1

 # echo 117 > /sys/class/gpio/export
 # echo out > /sys/class/gpio/gpio117/direction
 # cat /sys/class/gpio/gpio117/value
 # echo 1 > /sys/class/gpio/gpio117/value

- LED_2

 # echo 118 > /sys/class/gpio/export
 # echo out > /sys/class/gpio/gpio118/direction
 # cat /sys/class/gpio/gpio118/value
 # echo 1 > /sys/class/gpio/gpio118/value

- LED_3

 # echo 124 > /sys/class/gpio/export
 # echo out > /sys/class/gpio/gpio124/direction
 # cat /sys/class/gpio/gpio124/value
 # echo 1 > /sys/class/gpio/gpio124/value

- LED_4

 # echo 125 > /sys/class/gpio/export
 # echo out > /sys/class/gpio/gpio125/direction
 # cat /sys/class/gpio/gpio125/value
 # echo 1 > /sys/class/gpio/gpio125/value

17. Low Power Expansion GPIO

- Run the below command to control the Led

- LS_GPIO2_A

 # echo 42 > /sys/class/gpio/export
 # echo out > /sys/class/gpio/gpio42/direction
 # cat /sys/class/gpio/gpio42/value
 # echo 1 > /sys/class/gpio/gpio42/value
 # echo 0> /sys/class/gpio/gpio42/value

- LS_GPIO2_B

 # echo 43 > /sys/class/gpio/export
 # echo out > /sys/class/gpio/gpio43/direction
 # cat /sys/class/gpio/gpio43/value

 # echo 1 > /sys/class/gpio/gpio43/value
 # echo 0 > /sys/class/gpio/gpio43/value

- LS_GPIO3_C

 # echo 88 > /sys/class/gpio/export
 # echo out > /sys/class/gpio/gpio88/direction
 # cat /sys/class/gpio/gpio88/value
 # echo 1 > /sys/class/gpio/gpio88/value
 # echo 0 > /sys/class/gpio/gpio88/value

- LS_GPIO3_D

 # echo 84 > /sys/class/gpio/export
 # echo out > /sys/class/gpio/gpio84/direction
 # cat /sys/class/gpio/gpio84/value
 # echo 1 > /sys/class/gpio/gpio84/value
 # echo 0 > /sys/class/gpio/gpio84/value

- LS_GPIO2_E

 # echo 39 > /sys/class/gpio/export
 # echo out > /sys/class/gpio/gpio39/direction
 # cat /sys/class/gpio/gpio39/value
 # echo 1 > /sys/class/gpio/gpio84/value
 # echo 0 > /sys/class/gpio/gpio84/value

- LS_GPIO3_F

 # echo 85 > /sys/class/gpio/export
 # echo out > /sys/class/gpio/gpio85/direction
 # cat /sys/class/gpio/gpio85/value
 # echo 1 > /sys/class/gpio/gpio85/value
 # echo 0 > /sys/class/gpio/gpio85/value

- LS_GPIO2_G

 # echo 40 > /sys/class/gpio/export
 # echo out > /sys/class/gpio/gpio40/direction
 # cat /sys/class/gpio/gpio40/value
 # echo 1 > /sys/class/gpio/gpio40/value
 # echo 0 > /sys/class/gpio/gpio40/value

- LS_GPIO3_H

 # echo 86 > /sys/class/gpio/export

 # echo out > /sys/class/gpio/gpio86/direction
 # cat /sys/class/gpio/gpio86/value
 # echo 1 > /sys/class/gpio/gpio86/value
 # echo 0 > /sys/class/gpio/gpio86/value

- LS_GPIO3_I

 # echo 76 > /sys/class/gpio/export
 # echo out > /sys/class/gpio/gpio76/direction
 # cat /sys/class/gpio/gpio76/value
 # echo 1 > /sys/class/gpio/gpio76/value
 # echo 0 > /sys/class/gpio/gpio76/value

- LS_GPIO1_J

 # echo 77 > /sys/class/gpio/export
 # echo out > /sys/class/gpio/gpio77/direction
 # cat /sys/class/gpio/gpio77/value
 # echo 1 > /sys/class/gpio/gpio77/value
 # echo 0 > /sys/class/gpio/gpio77/value

- LS_GPIO3_K

 # echo 5 > /sys/class/gpio/export
 # echo out > /sys/class/gpio/gpio5/direction
 # cat /sys/class/gpio/gpio5/value
 # echo 1 > /sys/class/gpio/gpio5/value
 # echo 0 > /sys/class/gpio/gpio5/value

- LS_GPIO1_L

 # echo 3 > /sys/class/gpio/export
 # echo out > /sys/class/gpio/gpio3/direction
 # cat /sys/class/gpio/gpio3/value
 # echo 1 > /sys/class/gpio/gpio3/value
 # echo 0 > /sys/class/gpio/gpio3/value

18. CAN Interface demo

- CAN interface can be tested by communicating between two boards.
- Connect two boards with supplied CAN cable.
- Enable CAN in both the boards with below command with board's console.

  # ip link set can0 type can bitrate 125000;ifconfig can0 up

- Configure one board as receiver as below command.

  # candump can0 &

- Configure another board as sender and send data using below command

  # cansend can0 18FC2A00#0100000000000000

- Observe on receiver side board receiving data

19. NOR Flash demo

- Go to Board's console and create text file and write some data into it by below command

  # vi write.txt

- Once done writing save and quit the above file by below command.
  # <ESC><:><wq>
- Check for the Nor flash node by below command

  # ls -l /dev/mtd0

- Erase NOR flash using below command

  # flash_eraseall /dev/mtd0

- Write the created file into NOR flash using below command.

  # time dd if=write.txt of=/dev/mtd0

- Read from NOR flash from the same location

  # dd if=/dev/mtd0 of=read.txt

  # cat read.txt

- Compare the read.txt, it should be same as write.txt

- Please note that, when we write data, we write only a few bytes of data. However, when we read, we **read the whole partition** instead of the initial few lines. Due to that, we see junk characters in the place where we did not write anything. So user need to read the file at very first few lines using vim and verify its data.

20. Wi-Fi Demo

- Open terminal and follow the below commands to run Wi-Fi demo

  # ip link show wlan0

  3: wlan0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
  link/ether 00:25:ca:14:11:6c brd ff:ff:ff:ff:ff:ff

- Edit /etc/wpa_supplicant.conf file as per below in target board using vi

  # vi /etc/wpa_supplicant.conf

- Add following contains in /etc/wpa_supplicant.conf and save it.

ctrl_interface=/var/run/wpa_supplicant

ctrl_interface_group=0
update_config=1

network={
ssid="closenet"
scan_ssid=1
key_mgmt=WPA-PSK
psk="123456789"
}

- Turn on hotspot from mobile device or from Router.
- Change its ssid name /Network name to "**closenet"** and password to **"123456789"** (Create Close network). Make sure Security Type must be **WPA-PSK or WPA2-PSK.**

  Note: User can select any name (ssid) and password (psk) except any space or special character here and can change above wpa_supplicant.conf file accordingly. For example, ssid "**Anil's iPhone**" is not valid one.

 # wpa_supplicant -B -i wlan0 -c /etc/wpa_supplicant.conf
 # iw wlan0 link
 # ip address list wlan0
 # udhcpc -i wlan0 -n -q
 # ip address list wlan0
 # ip route show
 # ping <Any network ip >

- Response should be as per below logs

PING 192.168.43.1 (192.168.43.1) 56(84) bytes of data.
64 bytes from 192.168.43.1: icmp_seq=1 ttl=64 time=12.6 ms
64 bytes from 192.168.43.1: icmp_seq=2 ttl=64 time=10.6 ms
64 bytes from 192.168.43.1: icmp_seq=3 ttl=64 time=27.7 ms
64 bytes from 192.168.43.1: icmp_seq=4 ttl=64 time=6.34 ms
64 bytes from 192.168.43.1: icmp_seq=5 ttl=64 time=22.8 ms
64 bytes from 192.168.43.1: icmp_seq=6 ttl=64 time=8.26 ms

- The above ping response validates Wi-Fi's working state

21. QT Chemical Plant demo

Description:

This demo is the showcase for chemical plant and it demonstrate how we can control chemical plant through GUI using QT application. Here for reference we use three different

chemical liquids with **Tank1, Tank2 and Tank3**. In practical, they may be less or more. Accordingly, we can change our GUI application.

Mixture Tank (at bottom in image) contains the mixture of different chemical liquid (Tank1, Tank2, Tank3). We give heat to mixture for product output at desired temperature. **Mixture Tank Temperature** shows current Mixture Tank temperature with **Thermometer**. We also added **Mixture Tank Temperature Graph** at the top-left corner for graphical representation of Mixture Tank Temperature with reference to timescale. We also added **Mixture Tank FAN** to demonstrate Mixture Tank filling process. FAN is running when we filling into mixture tank and stop when filling is done and mixture tank is empty.

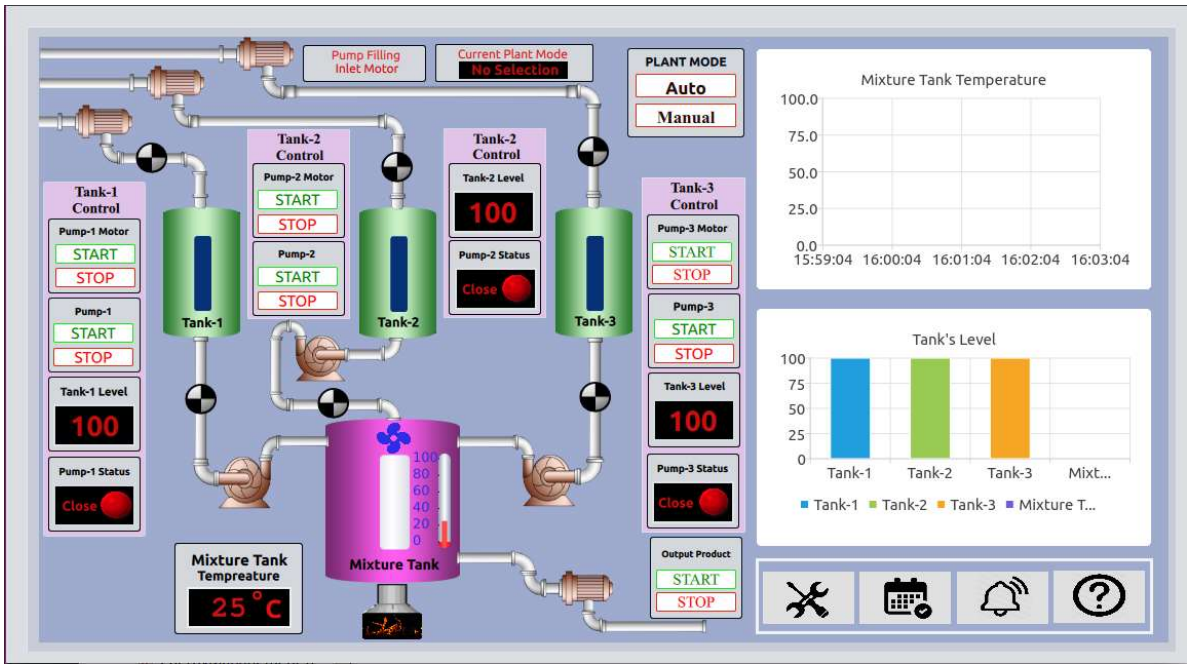

Figure 16 : QT Chemical Plant

Every Tank have its own **Tank Control Menu** to operate that tank. Each Tank Control Menu contains **Tank Level, Tank Status, Tank Filling Motor Control, Tank Pump Control** etc. **Pump Motor** or **Tank Filling Motor** is useful to fill liquid into respective Tank. **Pump Motor START** button will start fill that Tank and **STOP** button will stop filling it. When filling liquid is in progress respective **filters** (black/white circular shape) above the Pump will rotate and display progress. We also have **Pump START** and **STOP** button. It is useful for filling liquid from Tank to Mixture Tank. (It will empty respective Tank and fill into Mixture Tank.) **Filter** below pump will show progress for this.

Tank Level (inside control menu) display liquid level inside that Tank. Each Tank has its own Tank level and we can see all this level on **Tank's Level Graph.** Tank's Level Graph shows liquid level for each Tank as well as Mixture Tank level.

Tank Status (inside control menu) show status of the pump. Its shows **"Close"** status with **RED LED** when pump is not working (in action) and shows **"Open"** with **Green LED** when we run that pump to fill mixture tank.

We also added Control settings for **events, settings, alarms** and **helps**. In addition, there are two modes (**Auto Mode** and **Manual Mode**) to run the demo.

Steps to run Demo:

- Run command:
  # /usr/share/chemicalplanthmi-1.0/ChemicalPlantHMI
- Click on "**Auto**" Plant mode
  It will run demo in auto mode. In auto mode, all tanks liquid is falling into mixture tank. So first all tank level is decrease to zero gradually and mixture tank level is increasing. After all tanks are empty, we start filling them again and at the same time, we remove mixture tank product.
- Click on "**Manual**" Plant Mode.
  Now User has all access to start and stop pump manually.

  Based on mixture and all tank data, graphs data also changing in both modes.

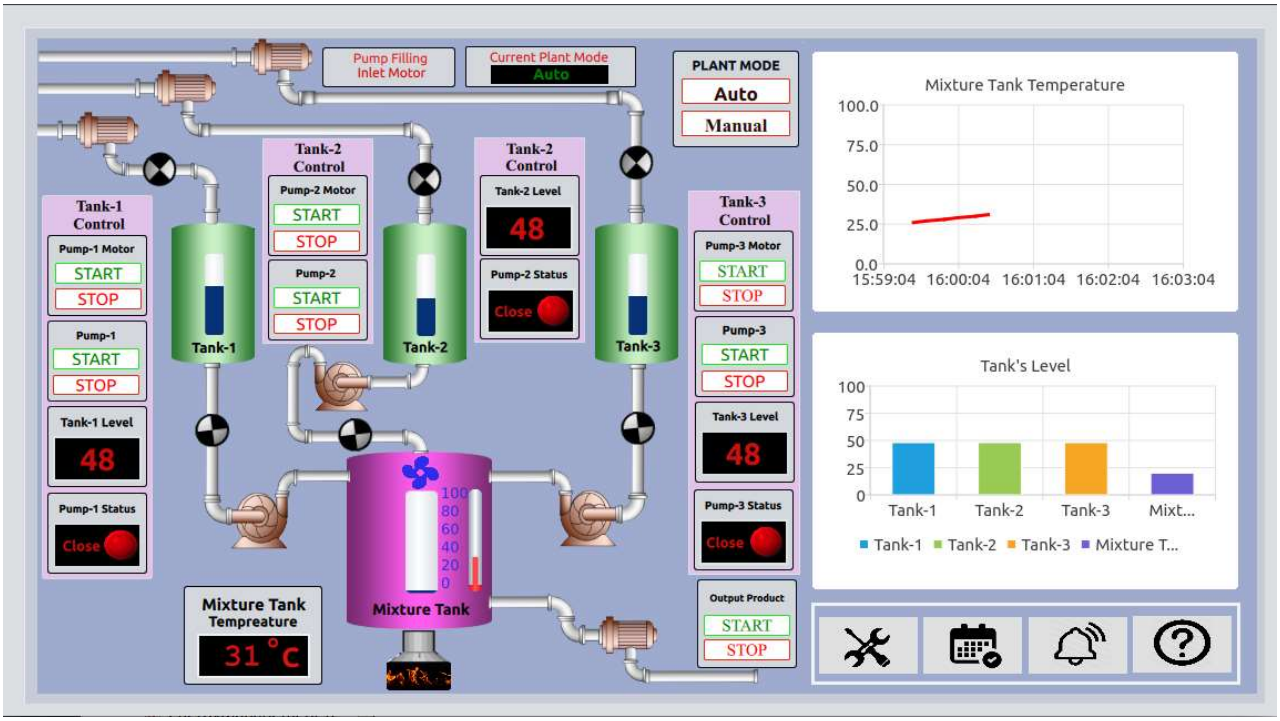

Figure 17 : Running Chemical Plant demo

As we can see in above figure 17 that, when we run demo in **auto** mode first Tank-1 level is decreasing (48 in above figure) and respectively mixture value is increasing.

22. QT Video HMI Application demo

Steps to run demo:

 # /usr/share/videohmiapp-1.0/VideoHMIApplication

Description:

This demo is the showcase for displaying any plant or production room's conditions and monitoring them. Here for reference we use four different production rooms with **Room-1, Room-2, Room-3 and Room-4**. In practical, they may be less or more. Accordingly, we can change our GUI application.

Figure 18 shows home screen of Video application. On default screen, we read real time room temperature and display on main screen. In addition, we capture same temperature log in **Temperature graph** as well with four different colors. (Shown at bottom)

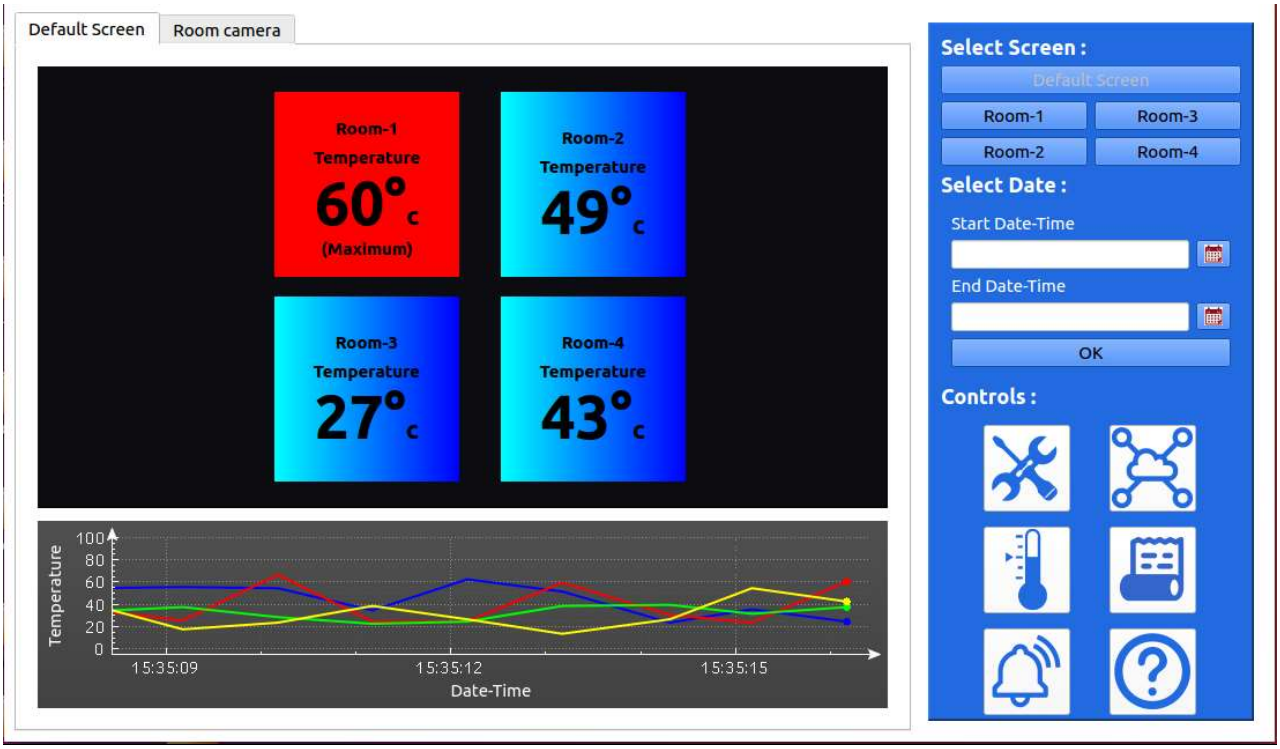

Figure 18 : HMI Video Application

We can also define temperature range here. For example, 0 to 60 where 0 is minimum and 60 is maximum acceptable temperature. When room's temperature goes beyond the acceptable range, demo app **will play the alarm sound** and room's temperature **notification label color change blue to red with maximum or minimum notification**. In figure 18, we can see that Room-1 temperature (60 C) is beyond maximum limit (50 C) so label color is RED.

Here we get Room temperature data from CSV file. However, in actual, we can get data from different temperature sensors. Therefore, we will get real time data.

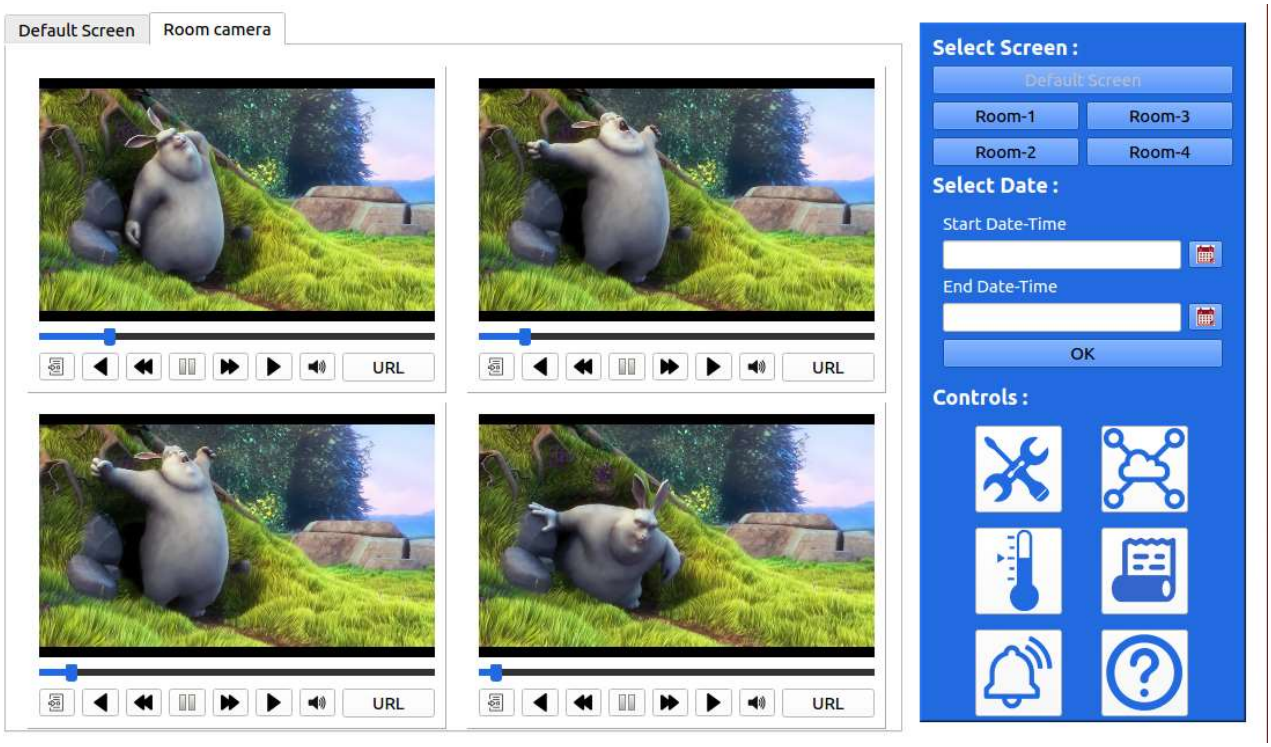

Figure 19 : Running multiple video files

As shown in below Figure 19 and figure 20, we can also run multiple video files and camera streaming on **Room Camera** Screen. For that first User need to click on Room Camera tab on top left corner. It will display **media player** through which we can open any video file. **Open** button (in media player) is useful to browse and open video file. After opening, we can play video by **PLAY** button (symbolic). We can also pause running video by **PAUSE** (symbolic) button. **URL** Button is useful for live streaming. User need to give **RTSP stream video** command to live streaming. (Figure 20)

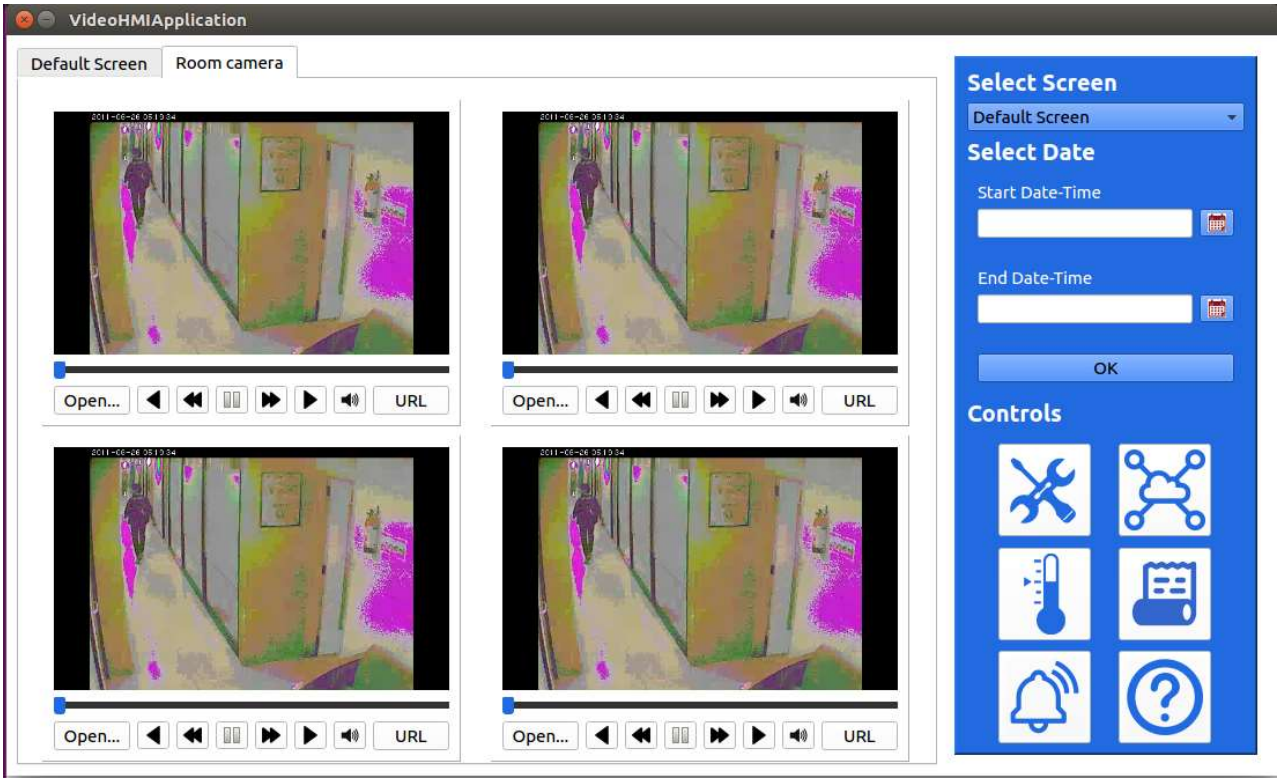

Figure 20 : Live Camera Streaming over RTSP

On Right – Top Corner, we have different Screen selection option. By default, default screen is selected, but user can go to particular camera screen to monitor that Room. For example, user can select Room-1 to monitor Room-1 condition. (Figure 21)

As shown in Figure 21, only Room-1 information is displayed. We can live stream or play video for room -1 only. Also temperature graph show only Room-1 data.

Select Date (on right top**) widget** is useful for displaying all room's temperature on graph based on between start date-time and end date time. User can select data and time by Clicking **Calendar** button just right after **start data-time** and **end date-time** text area. (Figure 22)
As shown in Figure 23, based on selected date and time we show **Temperature History graph.**

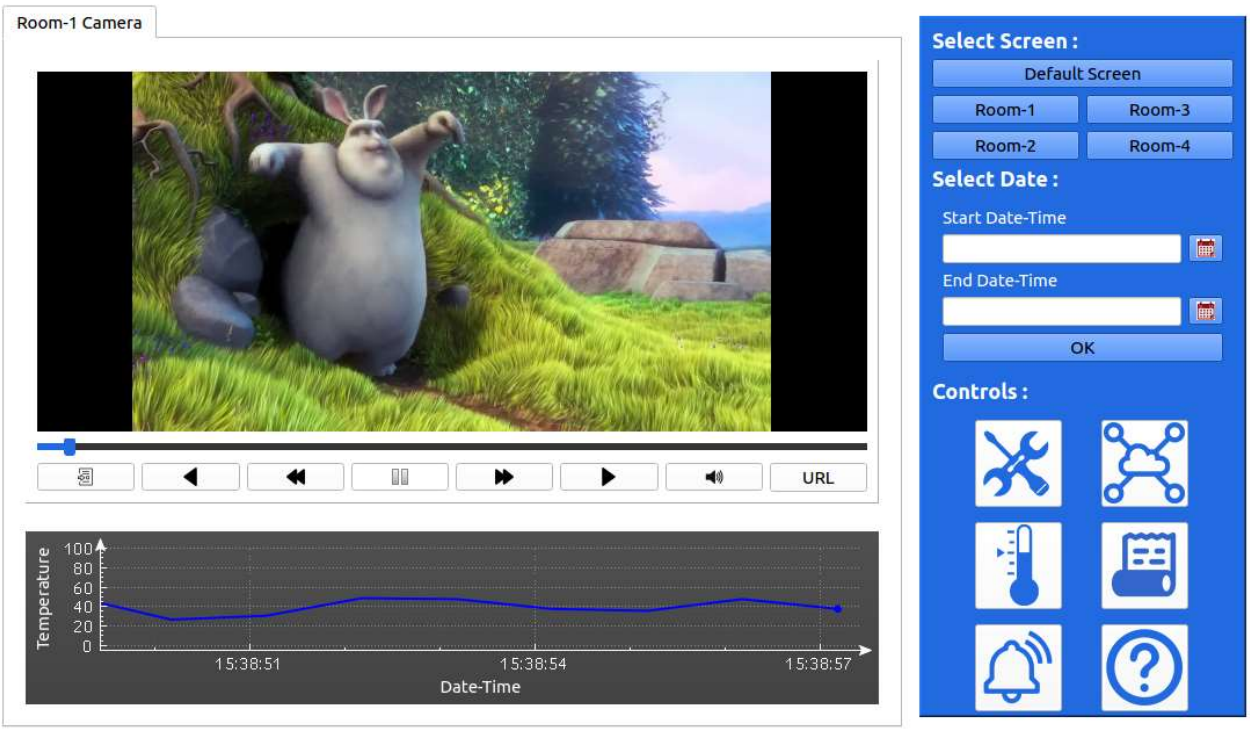

Figure 21 : Room-1 Screen

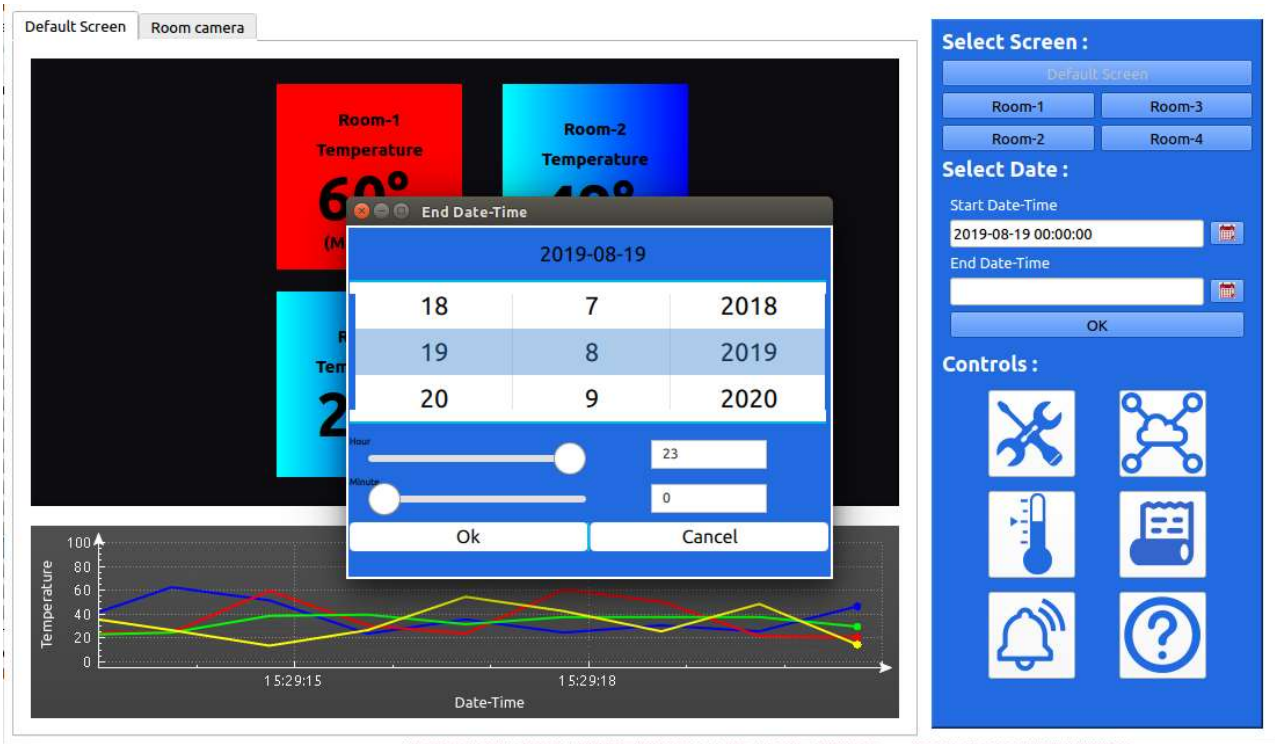

Figure 22 : Select DATE-TIME

Figure 23 : Temperature History based on selected time

We also have following **Controls** (at Right-bottom side):

1) Settings:
   Not implement.
2) Data on cloud
   Not implement.
3) Temperature Setting: (Figure 24)
   - Open the room's temperature setting pop-up window.
   - It is useful for set minimum or maximum room's temperature based on selection of rooms.
4) Data History: (Figure 25)
   - Open the window with table and graph.

   Data History Table:
   Display minimum or maximum temperature of the rooms with date and time.

Figure 24 : Room Temperature Limit setting

Figure 25 : Data History Screen

Data History graph:
Display history of all the room's temperature on the graph.
5) Alarm
Not implement
6) Helps

NOTE: Some of the QT features (like calendar widget closing) are not working fine in board due to X-WAYLAND platform dependency. It is QT's limitation that it will not work fine in WAYLAND and needed EGLFS Platform. In addition, QT media player video play sluggish. All this limitation not observed on x86 Machine with same QT code.

23. Alexa Demo

We have prepared Alexa demo to showcase Board ZigBee capability. For that we have add support for ZIGBEE service, apache2 with cgi-bin and SSL, mosquito, php. Kindly go through Alexa_User_Guide_v0.1 user guide.

Alexa_User_Guide_v0.1.docx

24. A2B Demo

Thor96 boards have A2B capability to connect multiple A2B slave devices. The current code demonstrates with one slave device to display A2B capability.

To perform A2B demo the pre-requisite is to have one EVAL-AD2428WB1BZ (Analog eval board), audio Aux cable, A2B Cable and headset.

Headset is connected to J10 connector on Thor96 board. A2B cable is connected between Thor96 board at J20 and EVAL-AD2428WB1BZ board on J7.

Audio source input (Mobile) is given to LINEIN port of EVAL_AD2428WB1BZ board using Aux cable. Aux cable is connected between mobile and LINEIN (J2) of EVAL-AD2428WB1BZ board.

Next, play the song on mobile and listen it on headphone. Please see figure 26 for more details.

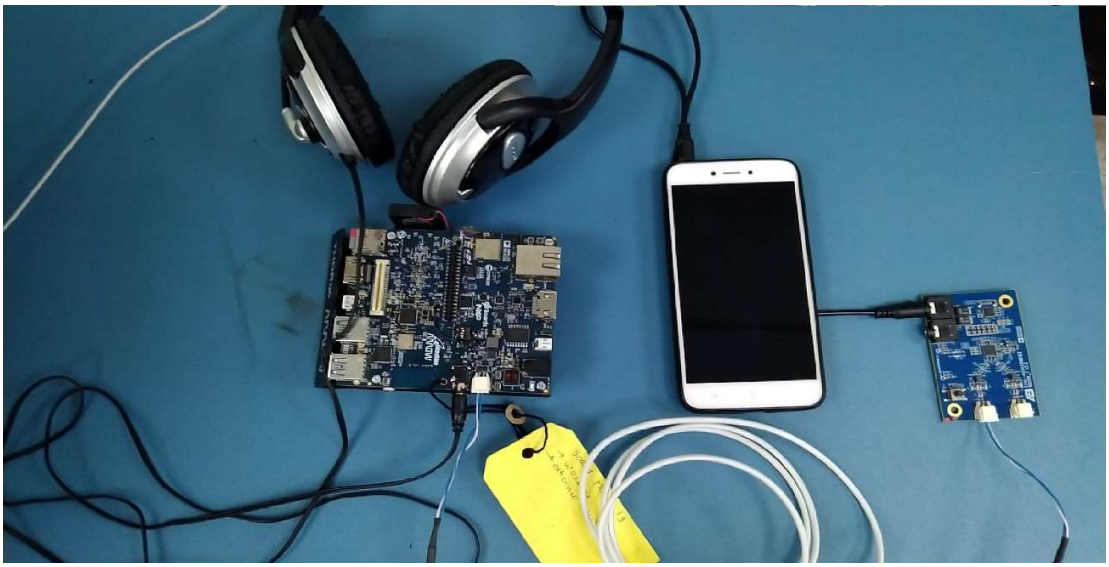

Figure 26 : A2B Demo Setup

- A2B source code is available in Thor96 board's home (/home/root/) directory.

  #cd ~/A2B

- Use below command to extract A2B.tar.gz tar file.

  # tar -xvzf A2B.tar.gz

The A2B.tar.gz folder contains Target, ReadMe.txt, 2019-10-09-LWSC-A2B Click through SLA.PDF and A2B_Application_code.patch.The Target folder contains the A2B full source code. Before you proceed to run or edit A2B source code you must agree with **2019-10-09-LWSC-A2B Click Thru SLA.PDF** Analog's license terms and conditions.

- Apply below command to shorten terminal console path

  # cd /home/root/A2B/A2B/Target/examples/demo/a2b-linux/a2b-adsp-sc584-linux/Makefiles

  # PS1='[\A]\u:\W>'

- Apply below command to enable the sync clock for A2B trans-receiver

  # arecord -Dplughw:1,0 -f S32_LE -r 48000 -c 2 | aplay -Dplughw:0,0 -f S32_LE -r 48000 -c 2 &

- Apply below command to discover the slave and start the playback path in network the slave code in network. Below path contains predefine binary.

  # ./staging/bin/a2bapp-linux –d

- Connect the audio source AUX cable in slave device's LINE_IN port and listen the sound on thor96 board's J10 (HP) port.

- Change to below directory to proceed with build process

  # cd /home/root/A2B/A2B/Target/examples/demo/a2b-linux/a2b-adsp-sc584-linux/Makefiles

- Apply below command to clean build and binary.

  #make -f Makefile-SC584.linux clean

- Apply below command to compile A2B source code and generate binary

  #make -f Makefile-SC584.linux all

- The generated binary is present in below path

  # cd /home/root/A2B/A2B/Target/examples/demo/a2b-linux/a2b-adsp-sc584-linux/Makefiles/staging/bin/

KNOWN ISSUES AND LIMITATIONS

- Please refer the Release note

CONTACT US

For any queries related to product, please contact us at tmain@arrow.com