Firmware User Manual

IMX8M THOR96 Reference Design

Date: January 19, 2022 | Version 4.0

CONTENTS

FIGURES

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
ML	Machine Learning
CNN	Convolutional Neural Network
ARM NN	ARM Neural Network
THOR96 board	Industrial Automation Board featuring the NXP i.MX 8M MPU
AES	Advanced Encryption Standard
AHAB	Advanced High Assurance Boot
AWS	Amazon Web Services
BSP	Board Support Package
CA	Certificate Authority
CAAM	Cryptographic Acceleration and Assurance Module
CMS	Cryptographic Message Syntax
CSF	Command Sequence File
CSR	Certificate Signing Request
CST	Code Signing Tool
DCD	Device Configuration Data
GG	AWS Greengrass
OS	Operating System
OTP	One-Time Programmable
PKI	Public Key Infrastructure
SA	Signature Authority
SCFW	SCU Firmware
SDP	Serial Download Protocol
SECO	Security Controller
SPL	Secondary Program Loader
SRK	Super Root Key
SSK	Security Starter Kit

TPM	Trusted Platform Module
USB	Universal Serial Bus
TLS	Transport Layer Security
RSA	Rivest–Shamir–Adleman
IoT	Internet of Things
HSM	Hardware Security Module
PKCS#11	PKCS#11 (Public Key Cryptography Standards) defines an API to communicate with cryptographic security tokens such as smart cards, USB keys and HSMs
HW	Hardware
MQTT	Message Queuing Telemetry Transport
SSL	Secure Sockets Layer
SHA	Secure Hash Algorithm
SDK	Software Development Kit
ECC	Elliptic Curve Cryptography
ARN	Amazon Resource Name
SECO	Security Controller
FW	Firmware
NV RAM	Non Volatile Random Access Memory
API	Application Programming Interface
UUID	Universally Unique Identifier
SCP	Secure Copy Protocol
SCU	System Control Unit
IAM	AWS Identity and Access Management
TSS	TPM2 Software Stack

1 INTRODUCTION

1.1 Purpose of the Document

Purpose of this document is to help developers flash firmware and demonstrate interfaces on iMX8M-THOR96 firmware.

1.2 About the System

This system is based on iMX8M processor and supporting multiple interfaces. This can facilitate for Human-Machine Interface experience.

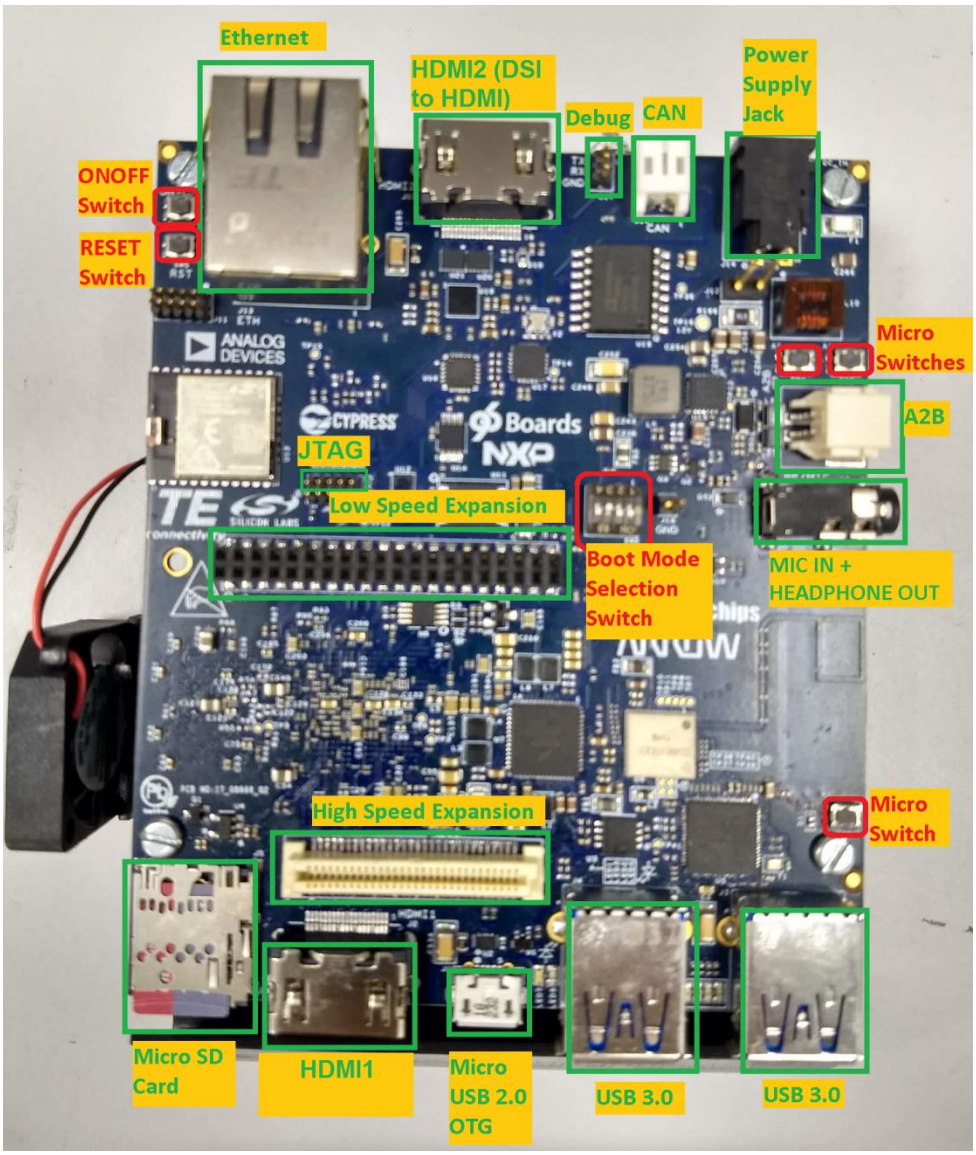

Figure 1: iMX8M RD THOR96 board

1.3 Intended Audience

This document is for developers and end-users who want to understand/flash/demonstrate interfaces on iMX8M-THOR96 firmware.

1.4 Prerequisites

Below are the list of hardware and software needed to demonstrate the interfaces on iMX8M-THOR96 Board:

- x86 host system having Linux Ubuntu 16.04 or 18.04 LTS installed (to build Yocto image)
- For building machine-learning components, at least 250 GB disk space is recommended
- Linux PC (Minicom for serial console, Optimal 16GB-RAM, Processor:Octa-core)
- Install Essential Yocto Project host packages
- Basic understanding of Linux commands
- Setup will require following:
  - THOR96 Board
  - SD-card -32GB
  - FTDI debug cable
  - Power Supply –
    - MEANWELL GST60A12-P1J
    - 5.5/2.1mm to 4.75/1.7mm cable DC plug converter
- Internet connectivity (Wi-Fi/Ethernet) of Board and Linux PC should be on same Network

2 ENVIRONMENT SETUP

2.1 Steps to build Yocto Image using build script.

A release package I.IMX8_Thor96 is available which contains build script, all the packages, BSP changes and the required patches for THOR96 firmware. User need to download release package first to build image for THOR96.

To build THOR96 firmware on LINUX HOST PC, user will follow below steps:

- Download new repo for THOR96 based on kernel version 5.10.52-hardknott release

```
$ sudo apt-get install repo
$ git config --global user.name "Your Name"
$ git config --global user.email "Your Email"
$ git config --global user.email "Your Email"
$ git config --list
```

- Download or clone release I.IMX8_Thor96, it contains as shown below:

```
I.IMX8_Thor96/Kernel_5_10_52
├── Software Docs
│   ├── Machine-Learning-Demos
│   │   └── ML_Demos_Guide_iMX8_L5_10_Rel_4_0.pdf
│   ├── ReleaseNote
│   │   └── ei_OnSemiCamModule_96B_Thor96_ReleaseNote.pdf
│   ├── TestReport
│   │   └── ei_OnSemiCamModule_96B_Thor96_Testcase.xlsx
│   └── UserGuide
│       └── ei_User_Guide_IMX8M_Thor96_L5_10_Rel_4_0.pdf
├── Thor96_L5_10_52_Rel_4_0_patches
├── yocto_build_setup_thor96.sh
└── Yocto_build_setup_steps.txt
```

- Run the build script to setup the Yocto environment on the LINUX based Host PC

```
$ cd I.IMX8_Thor96/Kernel_5_10_52/
$ sudo chmod 755  yocto_build_setup_thor96.sh
$ ./yocto_build_setup_thor96.sh
```

Note: [For building firmware image, it will take ~10 hours to download all packages and build, the time may vary based on your HOST PC configurations]

- After successful build the final SD Card image is available at below location:
  <root>/imx-yocto-bsp/build-xwayland-thor96/tmp/deploy/images/imx8mtho96/
- Filename should be imx-image-full-imx8mqthor96.wic.bz2 which is soft link of original build image file **imx-image-full-imx8mqthor96-<TIMESTAMP>.rootfs.wic.bz2**
- If user want to clean a previous build image and want to run it again then first clean it with command “cleanall” or “cleansstate”

```
$: bitbake imx-image-full -c cleanall
```

```
$: bitbake -v imx-image-full (If user want to turn on verbose)
```

- If user want to clean any package, then that can be done 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-scipy -c cleanall
$: bitbake python3-scipy     (Build scipy 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.

2.2 Download firmware package

- Download the provided SD Card (wic.bz2) image on Linux PC (host system)
- Open terminal in Host PC from left desktop panel or using keyboard shortcut (ctrl + t)
- From command terminal traverse to the location where firmware has been downloaded using cd command
- use ls command to verify the image downloaded
- Verify md5 check sum of downloaded image with given md5sum
- Extract the provided .bz2 image using bunzip2 command, which will take couple of minutes.
- Once done, will end with .wic image in the same directory and can again be verified using ls -l command. Steps are as following:

```
$ cd /home/user/download/imximages/
$ ls -l
$ md5sum <image name>.wic.bz2
$ bunzip2 -dk -f <image>.wic.bz2
$ ls -l
```

2.3 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 inside /dev directory
  *# ls -l /dev/sd**
- Open terminal and traverse to the location where downloaded firmware image is stored using cd command
- Ensure the extracted firmware image's file format is .wic using ls -l command
- Use below command for flashing, if the SD card's entry in Linux is /dev/sdX

```
$ sudo dd if=<Image>.wic of=/dev/sdX bs=1M conv=fsync status=progress; sync;
```

- Above command will take couple of minutes or more (depending upon PC config) to flash onto the SD card
- Once done remove and insert the SD card again, 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

2.4 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 will show drive like “E:”
- Download appropriate production image *.wic.bz2
- Extract *.wic.bz2 image using WinZip or 7-zip. It will create *.wic image.
- Run Win32 Disk Imager
- Select .wic image file and target drive i.e. E: for input Image File. (See below figure)

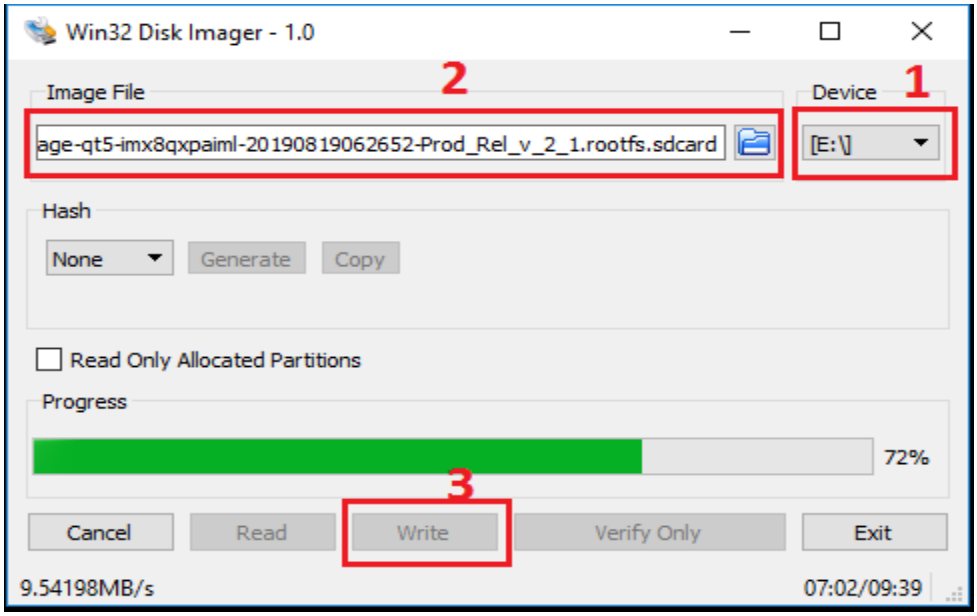

Figure 2: Win32 Disk Imager for flashing SD Card

- Click on Write
- After successful transfer, success message will pop up and there will be around 63.8 MB FAT partition. Inside this partition, one will find different DTB files and Image

2.5 Hardware Installation

- Place hardware board on a clean anti-static surface
- Insert flashed SD card to J5 SD card slot
- Attach serial cable's micro end to board's J10 Connector (near Ethernet connector) and USB end to host system i.e x86 PC's USB connector
- Attach Ethernet cable to board's Ethernet connector J12
- Provide 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

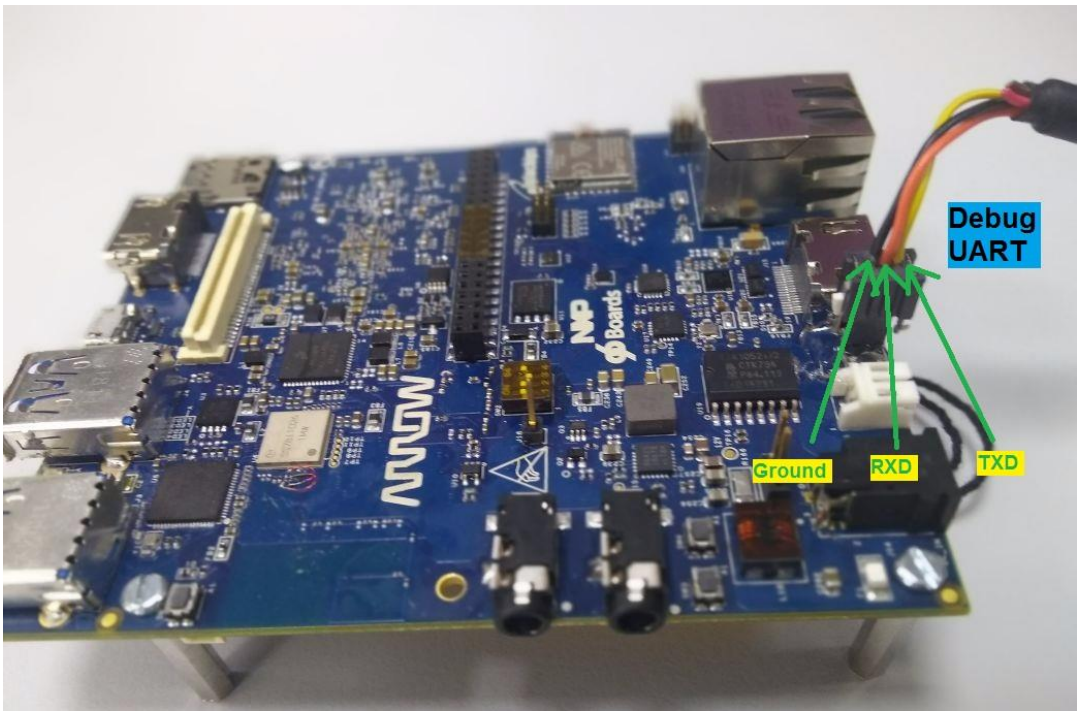

Figure 3: Thor96 – UART connection

2.6 Open board's terminal- console (minicom) on x86 host PC

- Ensure SD card is flashed, and serial cable is plugged-in to the board as described in hardware setup section.
- Attach serial cable's USB end to host x86 PC's USB.
- Ensure minicom is installed in x86 Ubuntu PC (host system)
- 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 powered-up, the above configured terminal will show logs on x86 and can interact with board using this open terminal

3 RUNNING DEMOS

Note: [Demos which require a change in .dtb file has been mentioned in their demo steps, else please keep the .dtb file as imx8mq-thor96.dtb]

3.1 Ethernet Demo

- Plug in ethernet cable into the target board as per above Figure.1.
- Power up the board.
- Once board gets booted, use the below command on console (minicom required)

```
root@imx8mqthor96:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr f2:b5:e5:f2:8f:8f
          inet addr:10.100.134.6  Bcast:10.100.135.255  Mask:255.255.252.0
          inet6 addr: fe80::f0b5:e5ff:fef2:8f8f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST DYNAMIC  MTU:1500  Metric:1
          RX packets:356 errors:0 dropped:98 overruns:0 frame:0
          TX packets:57 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:30213 (29.5 KiB)  TX bytes:8797 (8.5 KiB)

root@imx8mqthor96:~#
```

:

Figure 4 : Ethernet demo eth0 interface

- Above command will show the eth0 “IP” as per below screenshot. If IP is not showing up, check the ethernet Cable if it is plugged into the board.
- Ping <any server IP>

```
root@imx8mqthor96:~# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=118 time=16.6 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=118 time=16.5 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=118 time=16.5 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=118 time=16.5 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=118 time=16.5 ms
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 16.486/16.518/16.557/0.028 ms
root@imx8mqthor96:~#
```

Figure 5 : Ethernet demo running

3.2 HDMI1 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-1 connector as per Figure 1.
- Give power to board and go to terminal of x86 host system and open board's console as mentioned above.
- Console will show booting logs.
- Once booting is completed, console will stop at login prompt where user can enter username as root. (no password)
- At this moment, the connected HDMI display will show grey image of desktop and should stop complaining about "No Signal"

Figure 6 : HDMI 1 Display demo setup

3.2.1 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

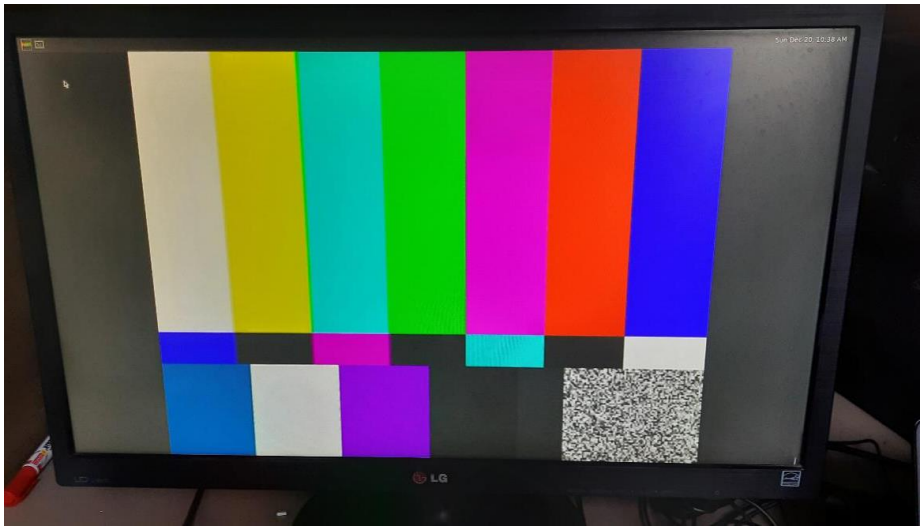

Figure 7: HDMI 1 Display test pattern output

3.2.2 Play local videos on HDMI display with audio

- Ensure Ethernet is connected with board.
- At board's console type below command from x86 minicom

```
$ ifconfig
```

- Get the IP address of Ethernet eth0 interface and note it down.
- Go to x86 host system and download sample mp4 video with audio.
- Go to video location from command line in x86 (no minicom required)
- Download any sample video file with .mp4 extension into your local PC
- Apply below command:

```
$ scp ./<file_name>.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 is copied using ls -l command. it will show Sample_Video_with_audio.mp4 in current directory.
- From the board's console, apply below command to play video over HDMI Display with audio (HDMI Display should have support of audio)
- Use below Command to check H/W card port

```
root@imx8mqthor96:~# aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: imxspdif [imx-spdif], device 0: S/PDIF PCM snd-soc-dummy-dai-0 [S/PDIF PCM snd-soc-dummy-dai-0]
Subdevices: 1/1
Subdevice #0: subdevice #0
card 2: adau1361audio [adau1361-audio], device 0: adau1361 adau-hifi-0 []
Subdevices: 1/1
Subdevice #0: subdevice #0
card 3: ad24xxa2bbus [ad24xx-a2b-bus], device 0: A2B24XX a2b24xx-hifi-0 []
Subdevices: 1/1
Subdevice #0: subdevice #0
card 4: imxaudiohdmi [imx-audio-hdmi], device 0: imx8 hdmi i2s-hifi-0 []
Subdevices: 1/1
Subdevice #0: subdevice #0
root@imx8mqthor96:~#

$ gst-launch-1.0 filesrc location=/home/root/<file_name>.mp4 ! decodebin name=dec ! videoconvert ! autovideosink dec. ! audioconvert ! audioresample ! alsasink device=plughw:4,0
```

Above command will print logs on console of board and will play video over HDMI display with audio. If HDMI Display doesn't support internal speaker, use audio jack.
In above command if "**alsasink device =plughw:4,0** ", make sure you have selected correct card and device number for **imxaudiohdmi** as shown in above command "aplay -l" response.

Figure 8 : HDMI1 Display video playback

3.3 HDMI2 (ADV7535) Demo

3.3.1 Play local videos on HDMI display

- Copy local <file_name>.mp4 videos using scp command to on board SD Card as mentioned in above demo.
- Insert HDMI cable into board's J15 HDMI2 connector
- Power up the board and go to terminal of x86 host system and open board’s console
- 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 imx8mq-thor96-lcdif-adv7535.dtb
$ saveenv
$ boot
```

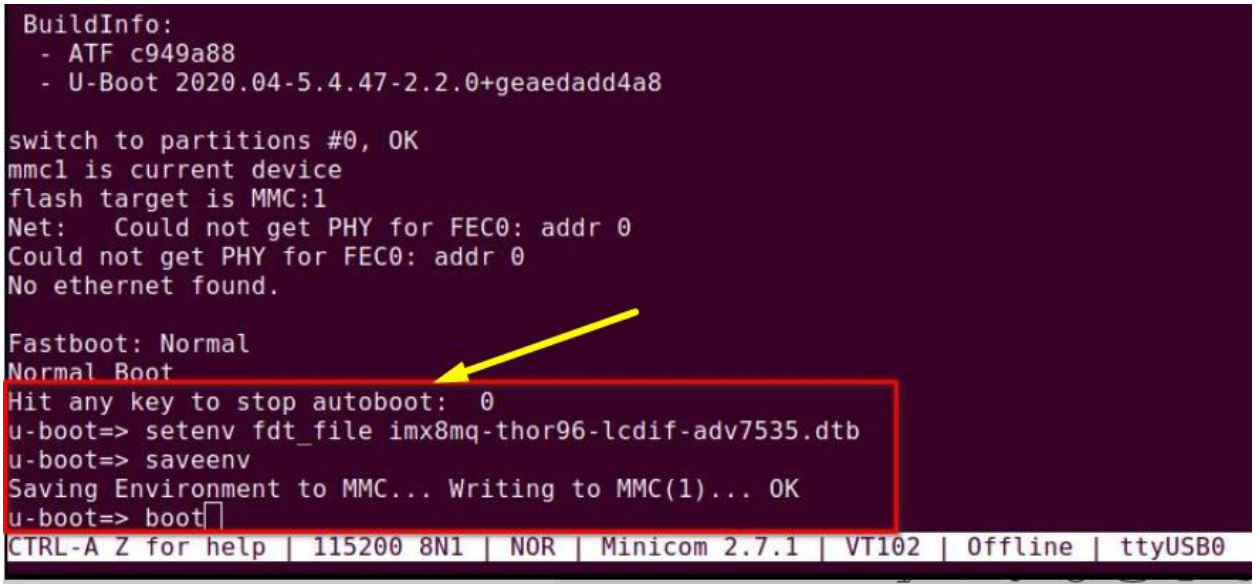

Figure 9 : U-boot console

- Console will show booting logs
- Once booting is completed, console will stop at login prompt where user can enter username as root. (no password)
- Start playback using below command

```
$ gst-launch-1.0 filesrc location=/home/root/test_30fps.mp4  typefind=true ! video/quicktime ! aiurdemux ! queue max-size-time=0 ! vpudec ! queue max-size-time=0 ! autovideosink
```

Figure 10 : HDMI 2 display video playback

3.4 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
- Power up the board and go to terminal of x86 host system and open board's console
- Apply dtb file changes as per below command on u-boot console

```
$ setenv fdt_file imx8mq-thor96-dual-display.dtb
$ saveenv
$ boot
```

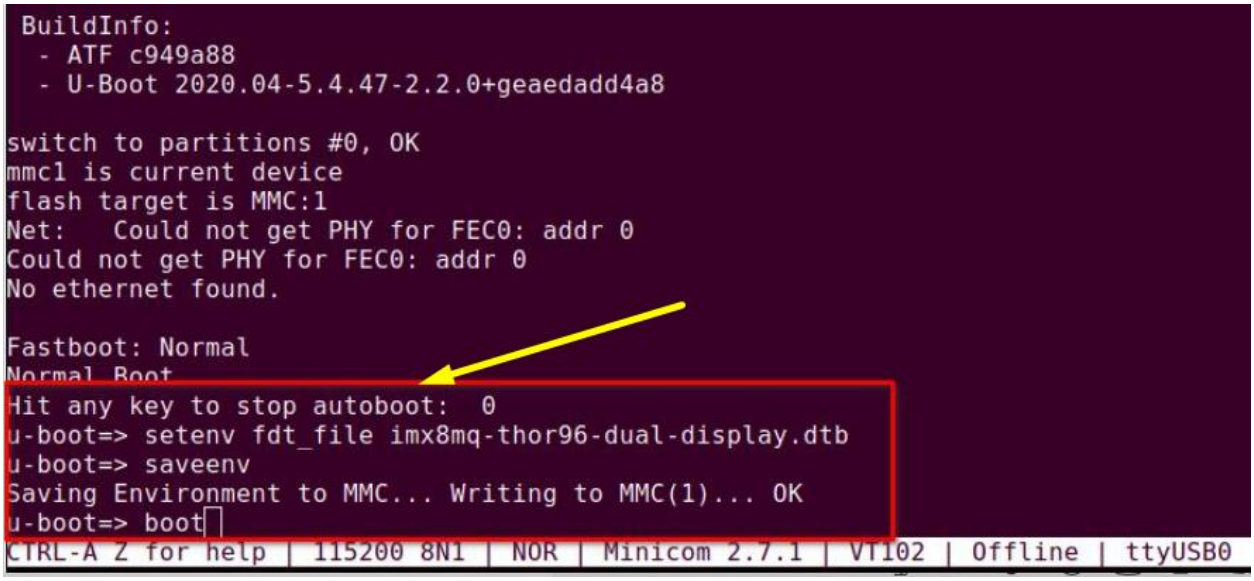

Figure 11 : U-boot console

- Console will show booting logs
- Login to board using root username with no password

3.4.1 Play local videos on HDMI display (Dual)

- Copy local sample_video.mp4 & sample_video2.mp4 or any .mp4 file videos using scp command to on board sd card as mentioned in above demo.
- Go to board's console (require minicom) and to ensure video is copied. Verify using ls -l command. It will show sample_video.mp4 in current directory.
- Apply below command to play video over HDMI(j2) Display

```
$ bitbake imx-image-full -c cleanall
$ bitbake -v imx-image-full (If user want to turn on verbose)
$ gplay-1.0 <file_name>.mp4 --video-sink="kmssink driver-name=imx-dcss" &
```

- Without much delay apply below command to play video over HDMI2 (j15) Display

```
$ gplay-1.0 <file_name>.mp4 &
```

Both connected HDMI will show dual video demo playback.

3.4.2 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

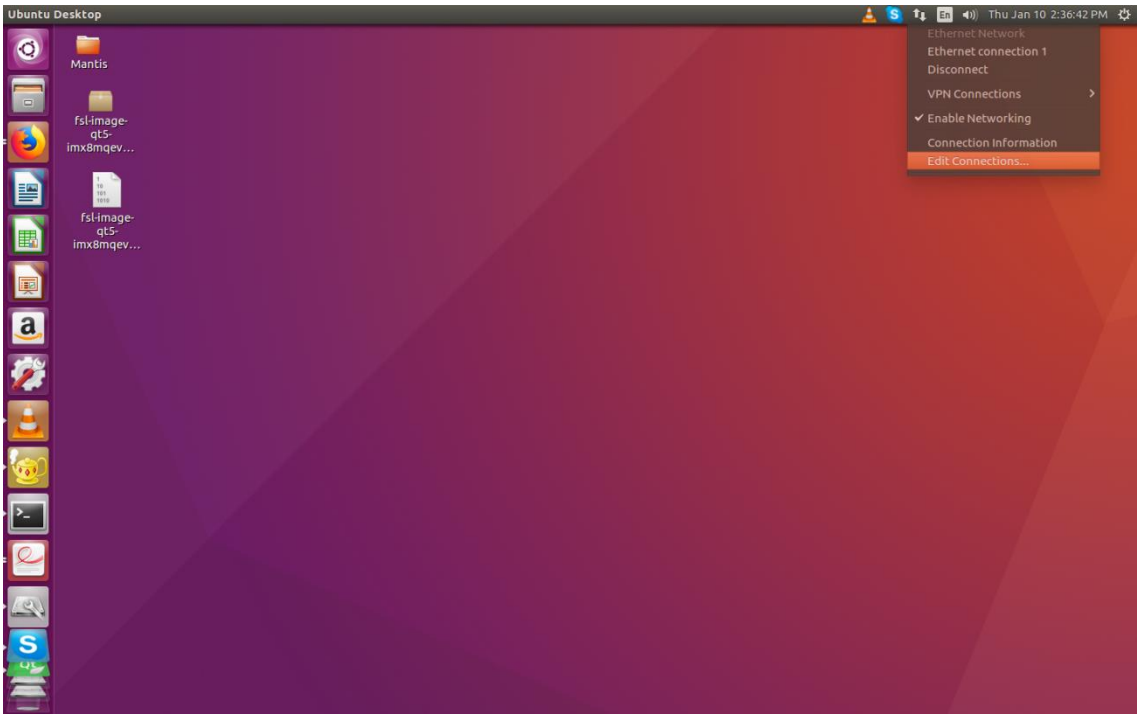

Figure 12: Edit network connection

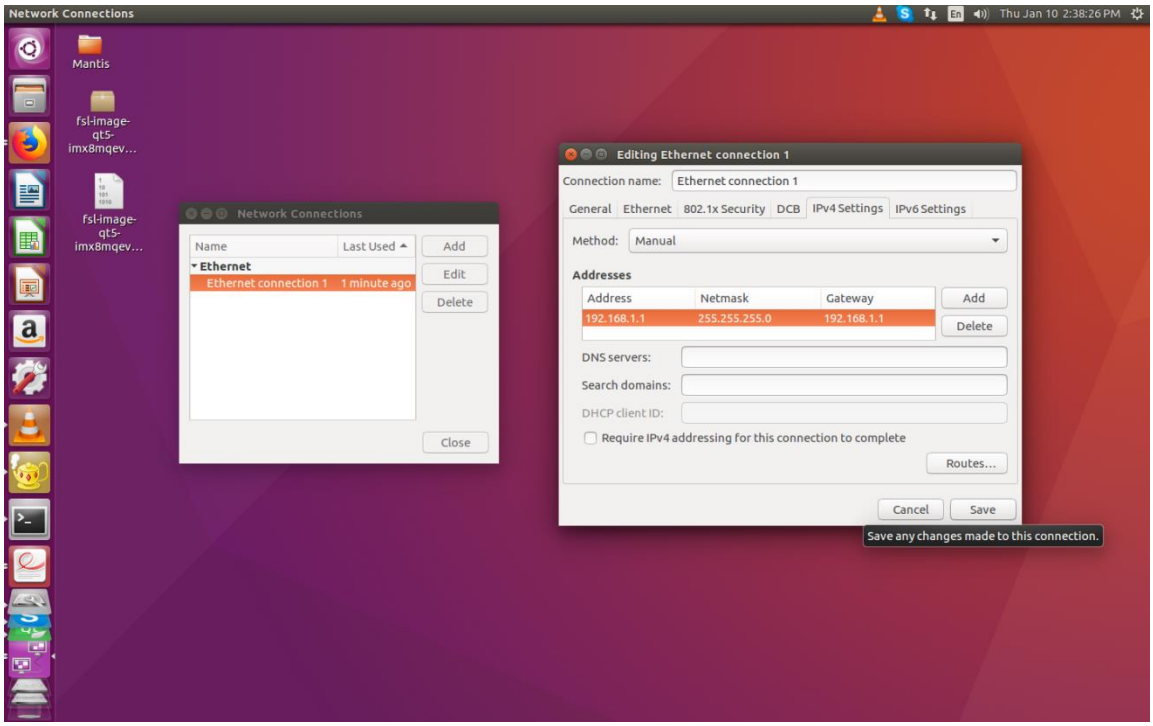

Figure 13: Create Static network

- 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
- Setup RTSP stream server on Linux PC as per below images
- Open VLC media player and start network stream

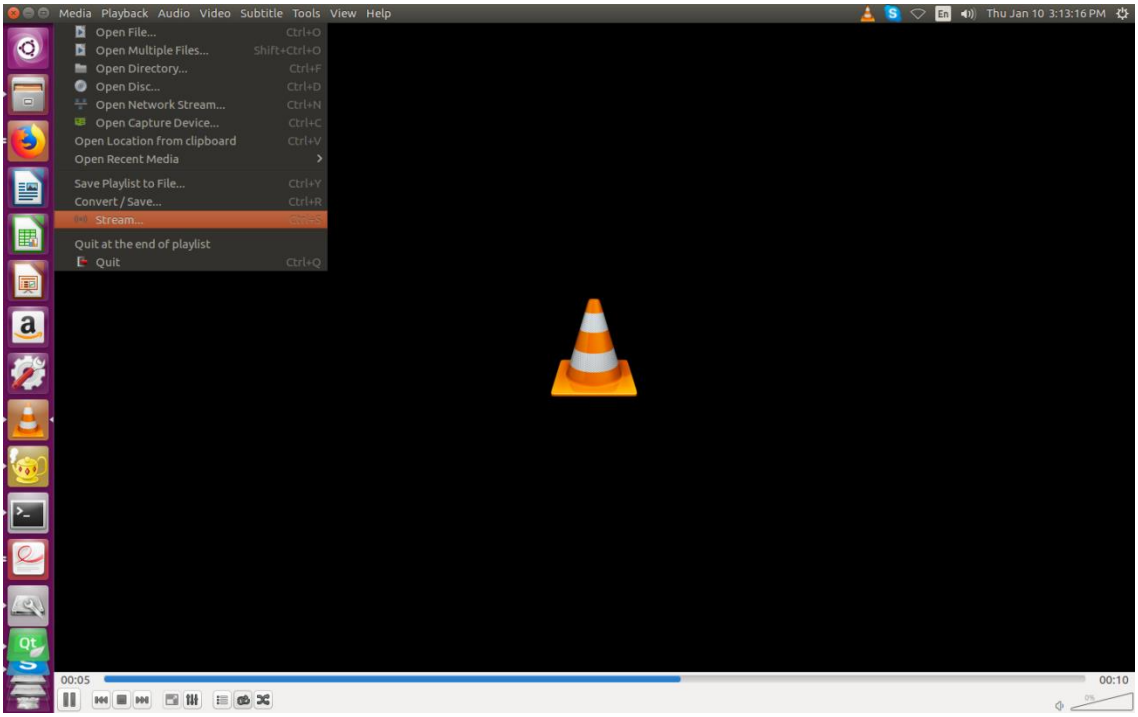

Figure 14 : Start Network Stream in VLC

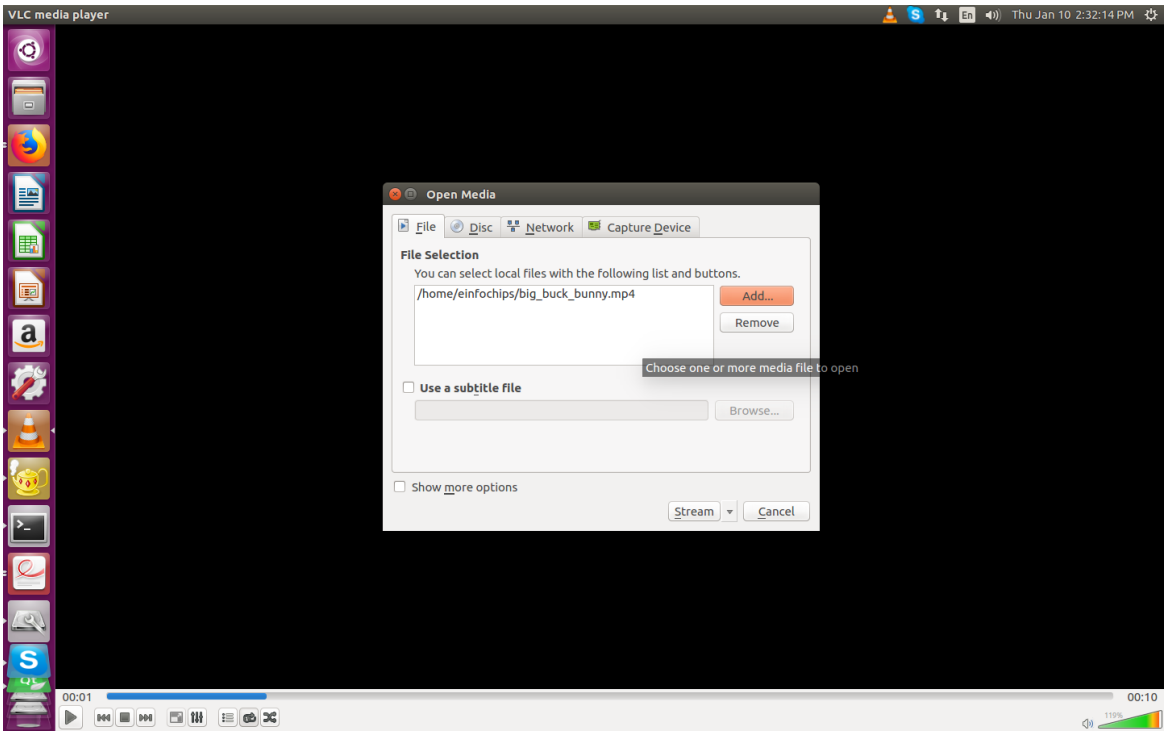

Figure 15 : Add Video file for RTSP stream

- Provide any video file that is to be streamed. i.e. big_buck_bunny.mp4

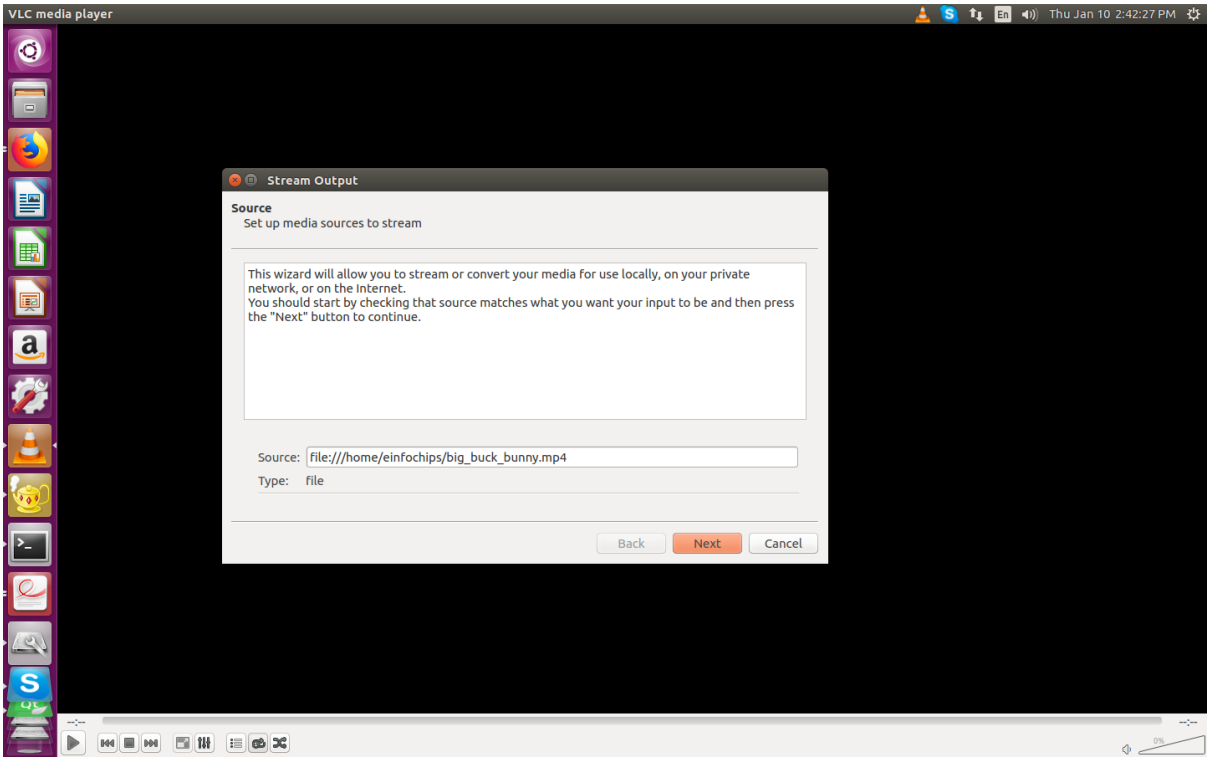

Figure 16 : Verify Video details

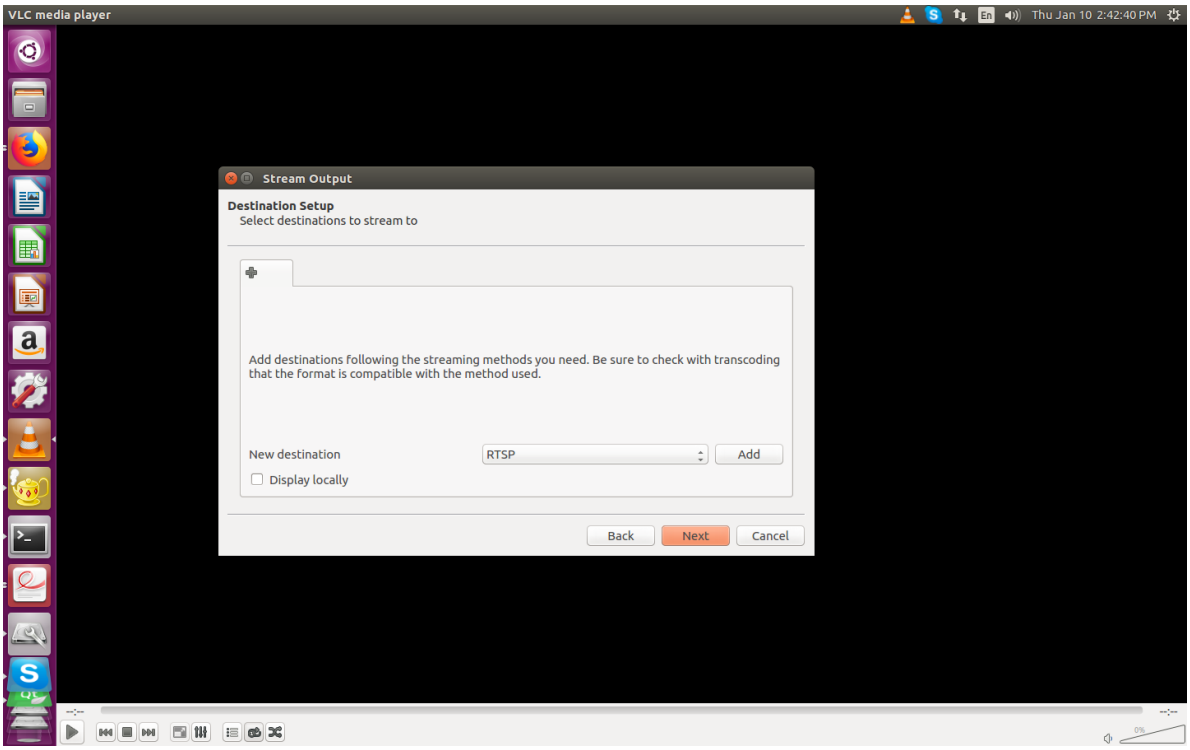

Figure 17 : Select RTSP option

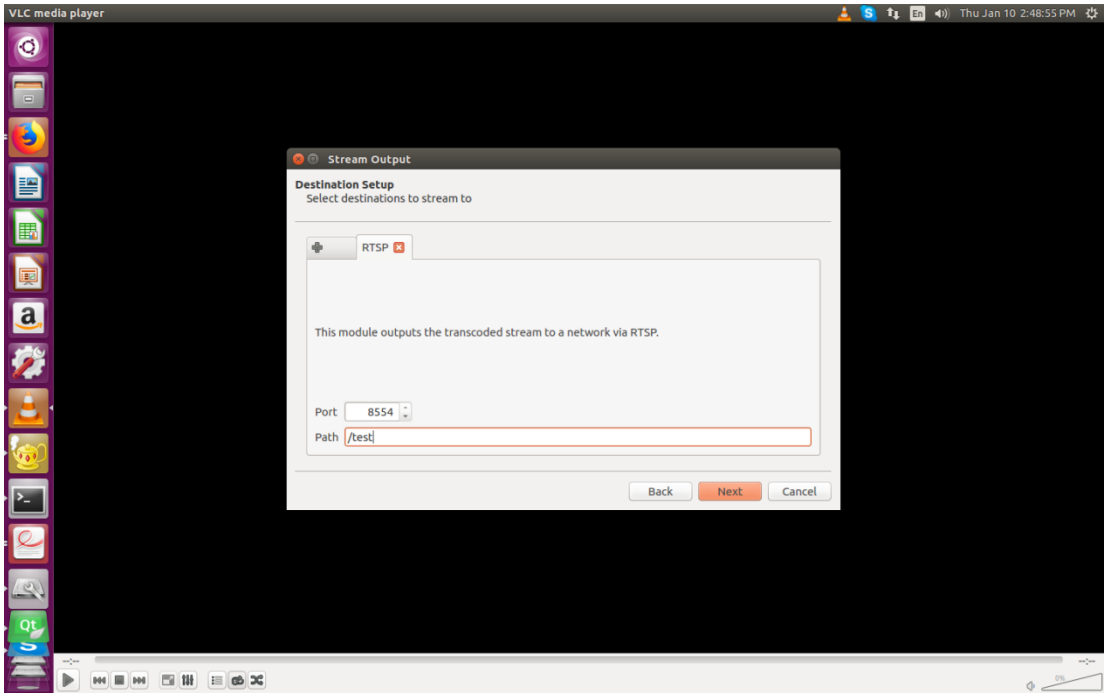

Figure 18 : Provide video stream name

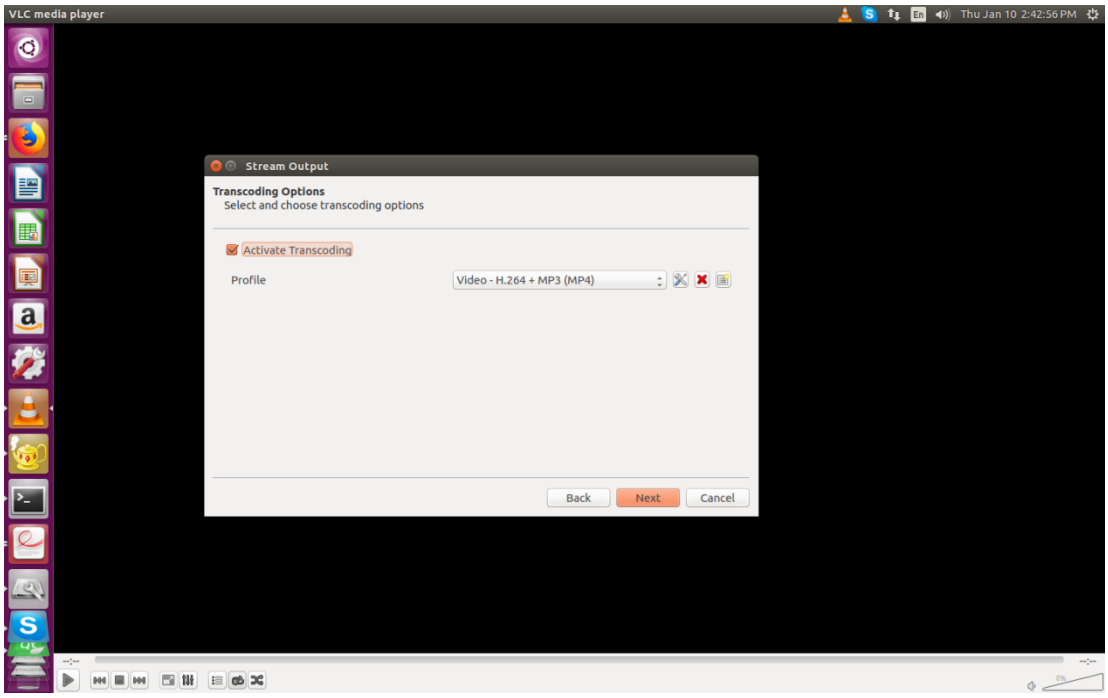

Figure 19 : Select Video codec

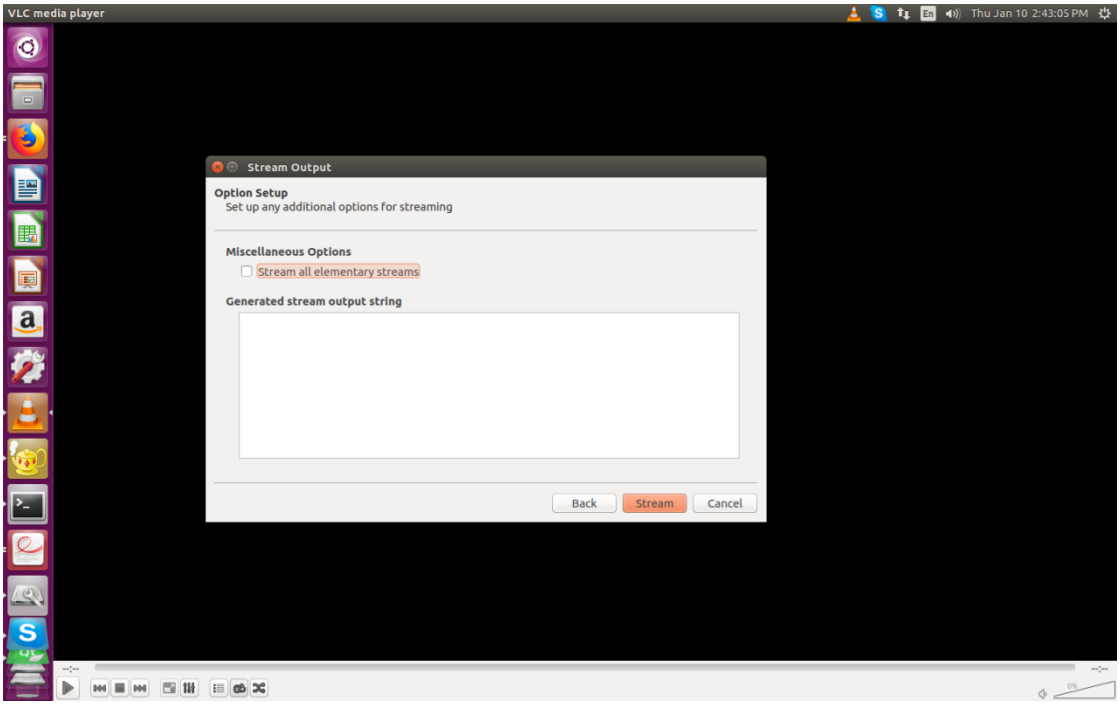

Figure 20 : Start Network Streaming

- After above steps VLC works as network stream and ensure that stream is in loop. Therefore, after finishing video play, it starts over again
- Now set static IP on board as well

```
$ ifconfig eth0 192.168.1.2
```

3.4.3 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, use below command to play the same stream over the network

```
$ gst-launch-1.0 rtspsrc location=rtsp://<IP Addr>:<Port No>/test ! decodebin ! autovideosink sync=true &
E.g. based on above setting for VLC and local LAN:
$ gst-launch-1.0 rtspsrc location=rtsp://192.168.43.85:8554/test ! decodebin ! autovideosink sync=true &
```

- Play simultaneously local video over HDMI1 (j2) using below command

```
$ gplay-1.0 <file_name>.mp4 --video-sink="kmssink driver-name=imx-dcss" &
```

3.5 Mezzanine DSI Display Demo

- Attach DSI display MX8_DSI_OLED to high-speed mezzanine connector

3.5.1 Play video on mezzanine DSI display

- After verification of 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  imx8mq-thor96-dcss-rm67191.dtb
$ saveenv
$ boot
```

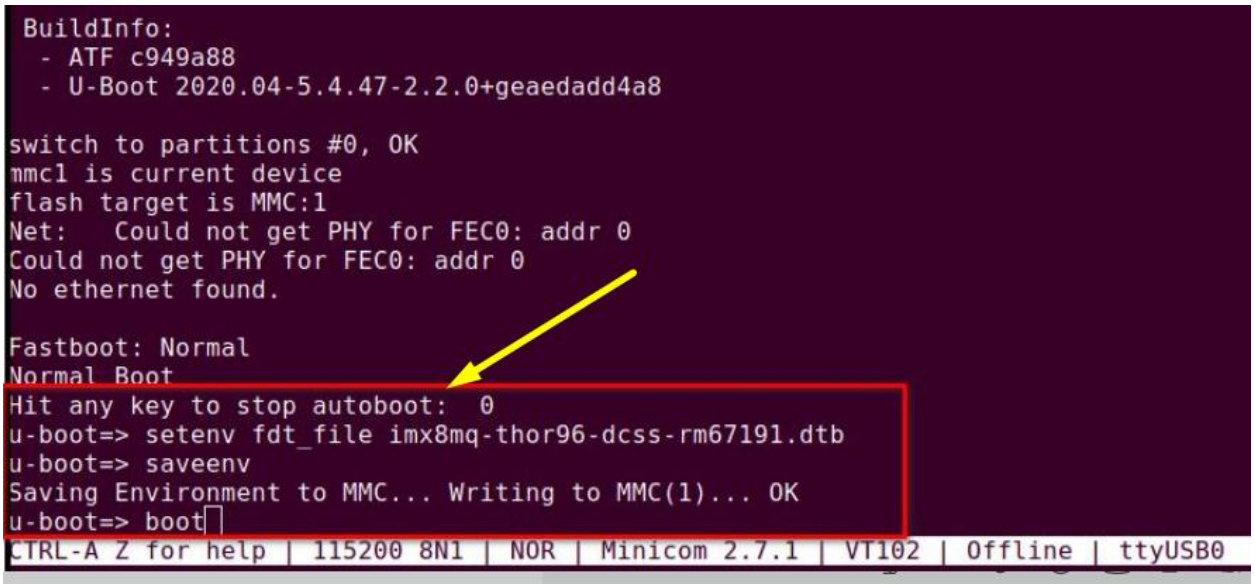

Figure 21 : U-boot console logs

- Use below command to showcase test pattern video on the OLED Display

```
$ gst-launch-1.0 videotestsrc ! autovideosink
```

Note: Mezzanine DSI display is not available on 96boards.org so it is not validated on Kernel 5.10 release But it will be validated with the steps mentioned as above.

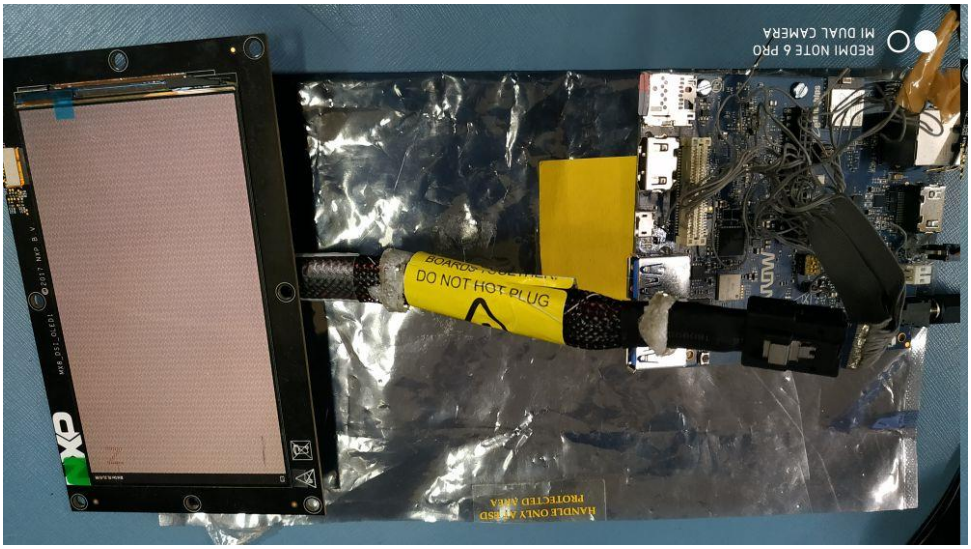

Figure 22 : iMX8M_Thor96 Platform Mezzanine DSI OLED

3.6 Camera Demo

3.6.1 Live stream from camera on HDMI display

- To watch live stream over the HDMI, connect HDMI1 Display.
- Attach D3-camera-mezzanine module, to high-speed expansion connector as shown in Figure
- Make sure the dtb environment is set with imx8mq-thor96.dtb using below command we can verify dtb environment.
- u-boot> printenv
- Power up the board
- If camera module is attached on CSI0 connector of the Attach D3-camera- mezzanine module.
- Then apply below command

```
$ gst-launch-1.0 v4l2src device=/dev/video0 ! video/x-raw,width=1280,height=720 ! autovideosink
```

- If camera module attached on CSI1 connector of the Attach D3-camera-mezzanine module.
- Then apply below command

```
$ gst-launch-1.0 v4l2src device=/dev/video1 ! video/x-raw,width=1280,height=720 ! autovideosink
```

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

3.6.2 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
- If camera module attached on CSI0 connector of the attached D3-camera- mezzanine module
- Apply below command to capture image from camera

```
$gst-launch-1.0 v4l2src device=/dev/video0 num-buffers=1 ! jpegenc ! filesink location=/home/root/test0.jpg
```

-
- If camera module attached on CSI1 connector of the attached D3-camera- mezzanine module
- Apply below command to capture image from camera

```
$gst-launch-1.0 v4l2src device=/dev/video1 num-buffers=1 ! jpegenc ! filesink location=/home/root/test0.jpg
```

- Above command will capture image named test0.jpg or test1.jpg in /home/root/ location
- Copy image from board to local PC using below command

```
$ scp test<CSI0 or CSI1>.jpg <username 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

3.7 Audio Codec Demo

- Power up the board and go to terminal of x86 host system and open board's console
- 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

```
$ u-boot> setenv fdt_file imx8mq-thor96.dtb
$ u-boot> saveenv
$ u-boot > boot
```

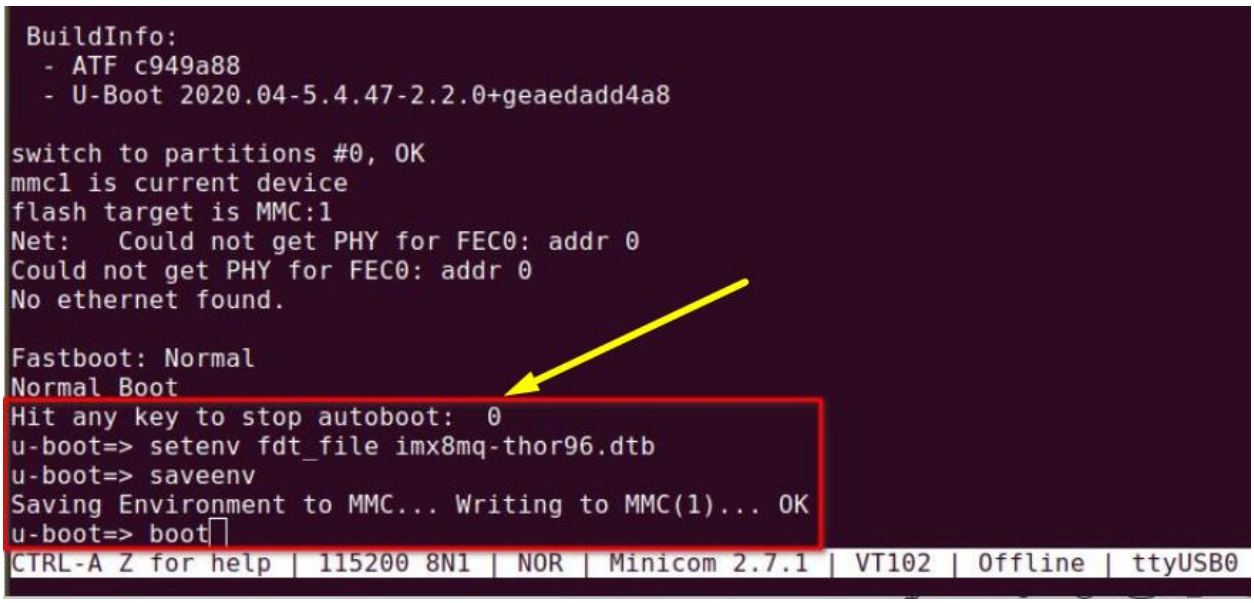

Figure 23 : U-boot console logs

- Power up the board
- Connect Handsfree with MIC to the J10 Connector as shown in Figure 1.
- 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

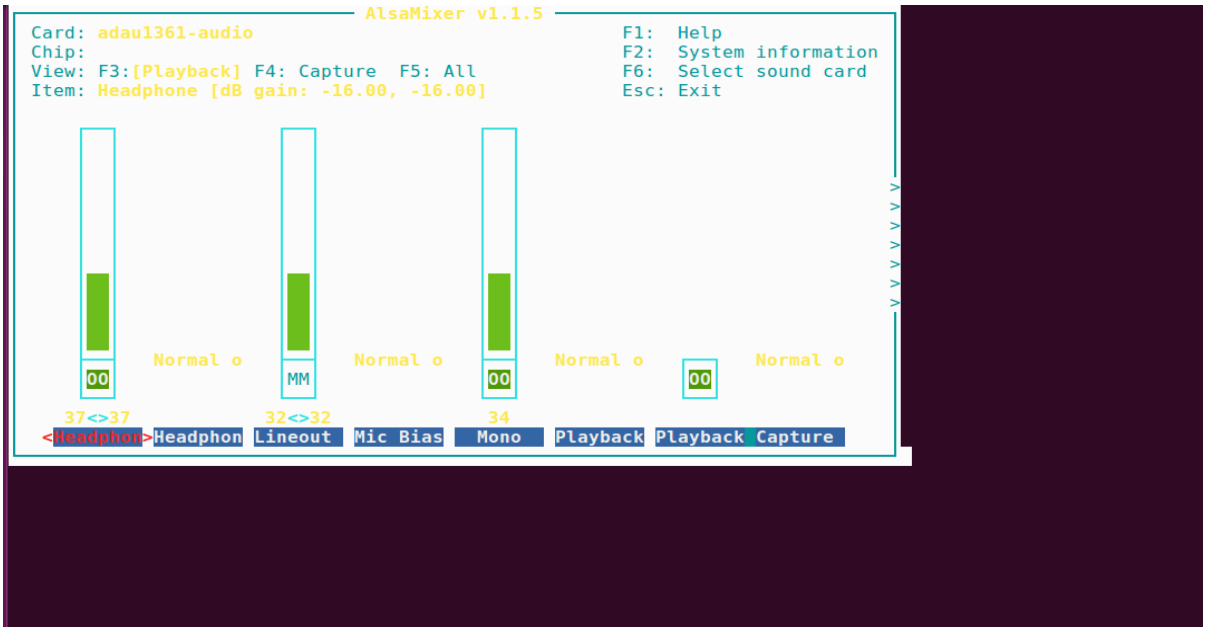

Figure 24 : Alsa Mixer Control Panel

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.

- 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 seen 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 settings, press ESC button to close alsamixer utility
- Enter below command to check the audio codec playback probed

```
$ sudo picocom -b 115200  -r -l /dev/ttyUSB0
$ aplay -l

# And find the below log from the list:
card 2:adau1361audio adau1361-audio],device 0: adau1361 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

```
card 2:adau1361audio [adau1361-audio],device 0: adau1361 adau-hifi-0 []
Subdevices: 1/1
Subdevice #0: subdevice #0
```

- To playback audio enter below command

```
$ aplay -Dplughw:2,0 SAI1/sample.wav
```

- To record audio enter below command

```
$ arecord -Dplughw:2,0 -f dat record.wav
```

- To test loopback of Handsfree MIC use below command

```
$ arecord -Dplughw:2,0 -f dat | aplay -Dplughw:2,0 -f dat
```

3.8 LTE Demo

- Go to board's console (require picocom) and make sure the dtb environment is set with imx8mq-thor96.dtb using below command we can verify dtb environment.
- u-boot>printenv
- Connect Quectel module with target board on the mezzanine connectors as following:
- Go to Board’s console and Initialize LTE module running following script:

```
$ cd LTE/
$ ./ lte_setup.sh
```

Figure 25 : LTE Demo Setup

- Set user SIM Card APN using below command.

Note: please identify your SIM card APN as per your service provider

```
$ export LTE_APN=<Your APN>
i.e export LTE_APN=airtelgprs.com
```

- 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 to check connection and IP address.

```
$ ifconfig ppp0
```

- Please confirm IP first, then use below command it should be able to ping to google.com

```
$ ping www.google.com -I ppp0
```

3.9 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 pen drive, observe status logs

```
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 pen drive, observe status logs

```
usb 2-1.2: USB disconnect, device number 4
```

3.10 USB OTG as Devices

- Please confirm the DTB environment, it should be imx8mq-thor96.
- Connect USB cable (same as 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
>Hello
>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

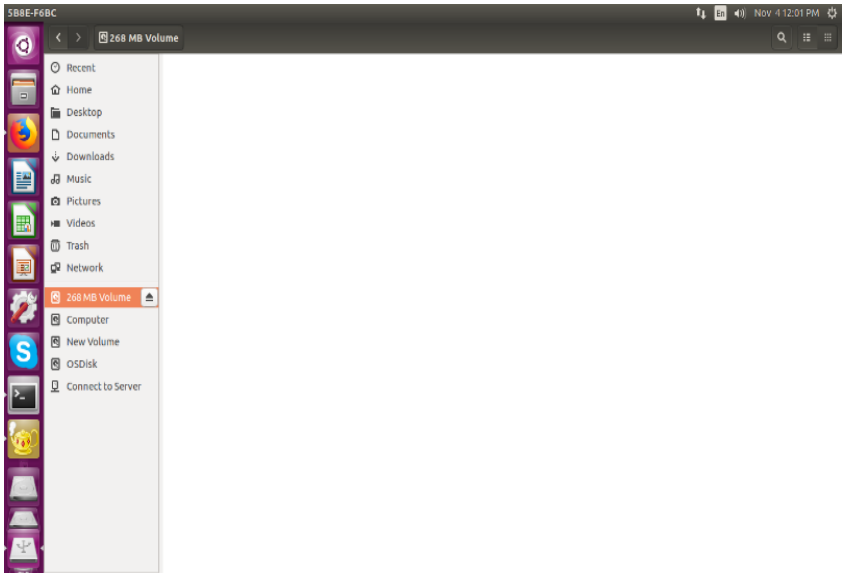

Figure 26 : USB Mass Storage on HOST system

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.

3.11 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 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, we get below status log

```
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 pen drive, we get below status log

```
usb 2-1.2: USB disconnect, device number 4.
```

3.12 Bluetooth

- Please confirm the dtb environment, it should be “imx8mq-thor96.dtb”.
- Please remove LTE Module if connected.
- 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 command response is completed

```
$ hciconfig hci0 up
```

- If Above command runs successfully, user can observe Blue LED glows on Board

```
root@imx8mqthor96:~# hciconfig hci0 -a
hci0: Type: Primary Bus: UART
BD Address: D4:53:83:0A:DB:D7 ACL MTU: 1021:8 SCO MTU: 64:1
UP RUNNING
RX bytes:1451 acl:0 sco:0 events:81 errors:0
TX bytes:1225 acl:0 sco:0 commands:81 errors:0
Features: 0xbf 0xfe 0xcf 0xfe 0xdb 0xff 0x7b 0x87
Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
Link policy: RSWITCH SNIFF
Link mode: SLAVE ACCEPT
Name: 'imx8mqthor96'
Class: 0x200000
Service Classes: Audio
Device Class: Miscellaneous,
HCI Version: 4.2 (0x8) Revision: 0x151
LMP Version: 4.2 (0x8) Subversion: 0x6119
Manufacturer: Broadcom Corporation (15)
root@imx8mqthor96:~#
```

- User will get the hci0 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

3.13 EEPROM

- Run below command to test EEPROM

```
$ echo eInfochips > /sys/bus/i2c/devices/i2c-1/1-0050/eeprom
$ cat /sys/bus/i2c/devices/i2c-1/1-0050/eeprom | hexdump -C
```

- User will be able to check below contents:

```
root@imx8mqevk:~# cat /sys/bus/i2c/devices/i2c-1/1-0050/eeprom | hexdump -C
00000000  65 49 6e 66 6f 63 68 69  70 73 0a 0a ff ff ff ff  |eInfochips.....|
00000010  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
00008000
```

3.14 USER LED

- Run the below command to control the LED
- USER_LED_1

```
# USER_LED_1 ON
$ echo 255 > /sys/class/leds/green\:user1/brightness

# USER_LED_1 OFF
$ echo 0 > /sys/class/leds/green\:user1/brightness
```

- USER_LED_2

```
# USER_LED_2 ON
$ echo 255 > /sys/class/leds/green\:user2/brightness

# USER_LED_2 OFF
$ echo 0 > /sys/class/leds/green\:user2/brightness
```

- USER_LED_3

```
# USER_LED_3 ON
$ echo 255 > /sys/class/leds/green\:user3/brightness

# USER_LED_3 OFF
$ echo 0 > /sys/class/leds/green\:user3/brightness
```

- USER_LED_4

```
# USER_LED_4 ON
$ echo 255 > /sys/class/leds/green\:user4/brightness

# USER_LED_4 OFF
$ echo 0 > /sys/class/leds/green\:user4/brightness
```

- BT_LED

```
# BT_LED ON
$ echo 255 > /sys/class/leds/blue\:bt/brightness

# BT_LED OFF
$ echo 0 > /sys/class/leds/blue\:bt/brightness
```

3.15 Low Power Expansion GPIO

- Run the below command to control the Led
- LS_GPIO2_A (PIN-23)

```
$ 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 (pin-24)

```
$ 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 (pin-25)

```
$ 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 (pin- 26)

```
$ 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 (pin- 27)

```
$ echo 39 > /sys/class/gpio/export
$ echo out > /sys/class/gpio/gpio39/direction
$ cat /sys/class/gpio/gpio39/value
$ echo 1 > /sys/class/gpio/gpio39/value
$ echo 0> /sys/class/gpio/gpio39/value
```

- LS_GPIO3_F (pin- 28)

```
$ 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 (pin- 29)

```
$ 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 (pin- 30)

```
$ echo 66 > /sys/class/gpio/export
$ echo out > /sys/class/gpio/gpio66/direction
$ cat /sys/class/gpio/gpio66/value
$ echo 1 > /sys/class/gpio/gpio66/value
$ echo 0> /sys/class/gpio/gpio66/value
```

- LS_GPIO3_I (pin – 31)

```
$ 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 (pin-32)

```
$ 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
```

- LS_GPIO3_K (pin-33)

```
$ 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_GPIO1_L (pin-34)

```
$ 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
```

3.16 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 is saved 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 data is written, it writes only a few bytes of data. However, if data is read, the whole partition is read instead of the initial few lines. Due to that, junk characters is seen in the place where no data is written. So, user need to read the file at very first few lines using vim and verify its data

3.17 Wi-Fi Demo

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

```
$ ifconfig wlan0 up
$ ip link show wlan0
```

- Above command will show the below details of 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 editor

```
$ vi /etc/wpa_supplicant.conf
```

- Add following contains in /etc/wpa_supplicant.conf file 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.

- Use below command to connect with your SSID

```
$ wpa_supplicant -B -i wlan0 -c /etc/wpa_supplicant.conf
```

- Now verify connection using below command

```
root@imx8mqthor96:~# iw wlan0 link
Connected to 8a:a3:03:7d:26:5e (on wlan0)
SSID: hello
freq: 2412
RX: 342 bytes (2 packets)
TX: 2442 bytes (17 packets)
signal: -34 dBm
rx bitrate: 1.0 MBit/s
```

```
tx bitrate: 24.0 MBit/s
bss flags: short-slot-time
dtim period: 2
beacon int: 100
root@imx8mqthor96:~# ip address list wlan0
```

- Discover IP address using below command

```
$ udhcpc -i wlan0 -n -q
```

- List inet address using below command

```
root@imx8mqthor96:~# ip address list wlan0
4: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen
1000
link/ether d4:53:83:0d:1b:55 brd ff:ff:ff:ff:ff:ff
inet 192.168.43.81/24 brd 192.168.43.255 scope global wlan0
valid_lft forever preferred_lft forever
root@imx8mqthor96:~#
```

- Ping any network IP and Response should be as per following logs:

```
$ ip route show
$ ping <Any network ip >

root@imx8mqthor96:~# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=118 time=49.6 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=118 time=105 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=118 time=87.9 ms
```

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

3.18 CAN Interface demo

- CAN interface can be tested by communicating between two boards.
- Connect two boards with supplied CAN cable

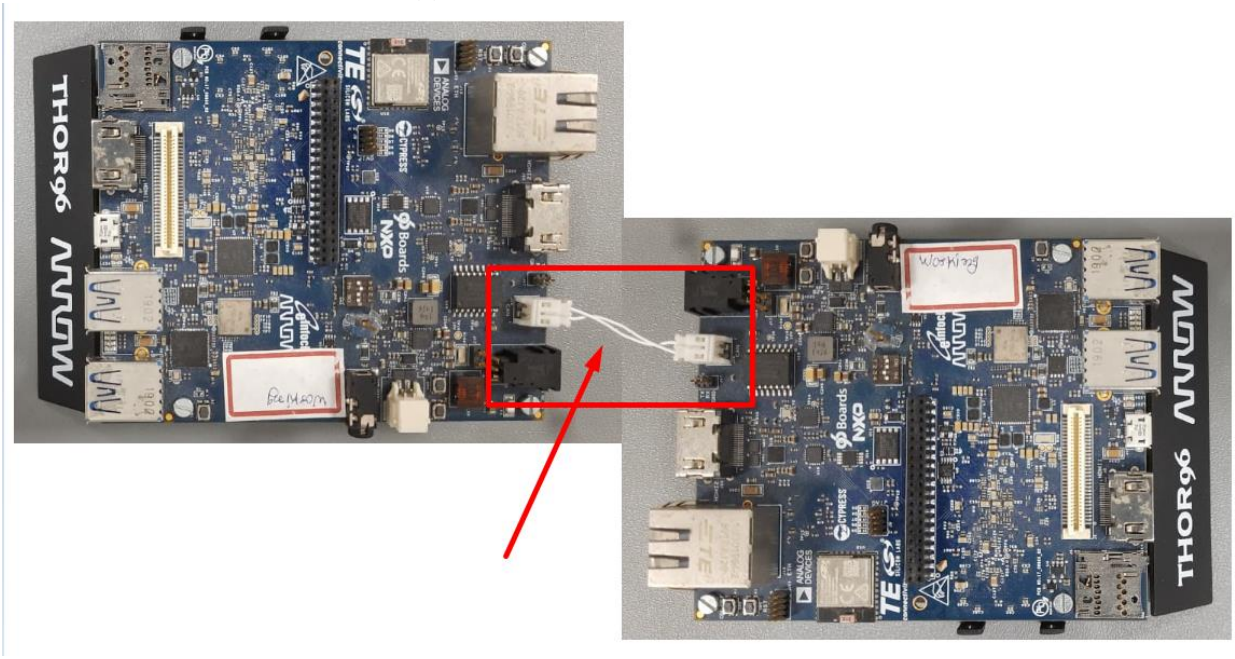

Figure 27 : CAN Demo setup

- 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 using below command.

```
$ candump can0 &
```

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

```
$ cansend can0 18FC2A00#0100000000000000
```

- Observe on receiver sideboard receiving data

3.19 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 to it on headphone. Please see figure 26 for more details.

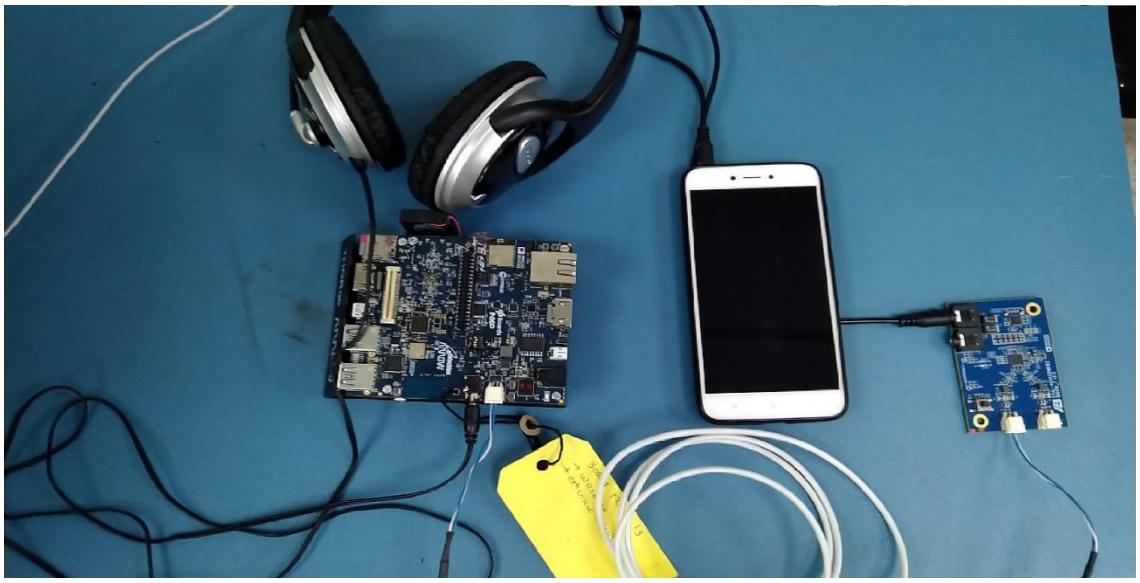

Figure 28 : A2B Demo Setup

- A2B source code is available in Thor96 board's home (/home/root/) directory
- Use below command to extract A2B.tar.gz tar file.

```
$ cd ~/A2B
$ tar -xzvf 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.*

- Use 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>'
```

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

```
$ arecord -Dplughw:3,0 -f S32_LE -r 48000 -c 2 | aplay -Dplughw:2,0 -f S32_LE -r 48000 -c 2 &
```

- Use 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
```

- Use 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/

3.20 Zigbee Demo

3.20.1 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.

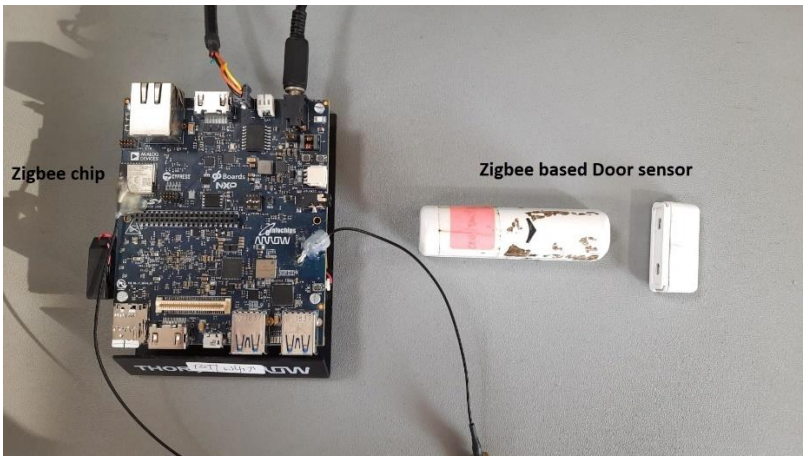

Figure 29 : Zigbee demo setup

3.20.2 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

```
$: bitbake imx-image-full -c cleanall
$: bitbake -v imx-image-full (If user want to turn on verbose)
```

 # chmod 777 Z3GatewayHost

- Ensure "zigbee" service is disabled. 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 start broadcasting, will connect above network, and on host side below log will appear.

```
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[]
```

3.21 Thread Demo

3.21.1 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 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.

3.21.2 Steps to test thread as below:

Server Host application:

- Change 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 there. 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]
[016302075468726561640000F00280001810315446563203230203230313820313 63A35303A353800]
[2018-12-05        13:41:19.037        +                0.001]        [driver->app        ]        [MGMT]
[016302075468726561640000F00280001810315446563203230203230313820313 63A35303A353800]
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]
[01637A10636C69656E742F7365727665720000000 8FDE185ACB16E000008E185ACB16E6C0AE8DBBC1A
020305081EAD1EFEFF9FFD90085B5]
                        [driver->app                                    ]                    [MGMT]
[01637A10636C69656E742F7365727665720000000 8FDE185ACB16E000008E185ACB16E6C0AE8DBBC1A
020305081EAD1EFEFF9FFD90085B509BE6]
[2018-12-05      13:41:19.081      +                0.001]      [NCP->driver      mgmt      ]      [MGMT]
[01637208E185ACB16E6C0AE810636C69656E742F7365727665720000000 8FDE185ACB16E0000DBBC1A
081EAD1EFEFF9FFD90085B509BE6A]
                        [driver->app                                    ]                    [MGMT]
[01637208E185ACB16E6C0AE810636C69656E742F7365727665720000000 8FDE185ACB16E0000DBBC1A
081EAD1EFEFF9FFD90085B509BE6A7EC18]
[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]
[01637A10636C69656E742F7365727665720000000 8FDE185ACB16E000008E185ACB16E6C0AE8DBBC1A
020305081EAD1EFEFF9FFD90085B5]
                        [driver->app                                    ]                    [MGMT]
[01637A10636C69656E742F7365727665720000000 8FDE185ACB16E000008E185ACB16E6C0AE8DBBC1A
020305081EAD1EFEFF9FFD90085B509BE6]
```

```
[2018-12-05 13:41:19.083 + 0.001] [NCP->driver mgmt ] [MGMT]
[01637208E185ACB16E6C0AE810636C69656E742F7365727665720000008FDE185ACB16E0000DBBC1A
081EAD1EFEFF9FFD90085B509BE6A]
[driver->app ] [MGMT]
[01637208E185ACB16E6C0AE810636C69656E742F7365727665720000008FDE185ACB16E0000DBBC1A
081EAD1EFEFF9FFD90085B509BE6A7EC18]

[2018-12-05 13:41:20.563 + 1.480] [NCP->driver mgmt ] [MGMT]
[01637A10636C69656E742F7365727665720000008FDE185ACB16E000008E185ACB16E6C0AE8DBBC1A
020305081EAD1EFEFF9FFD90085B5]
[2018-12-05 13:41:20.564 + 0.001] [driver->app ] [MGMT]
[01637A10636C69656E742F7365727665720000008FDE185ACB16E000008E185ACB16E6C0AE8DBBC1A
020305081EAD1EFEFF9FFD90085B509BE6]
[2018-12-05 13:41:20.567 + 0.003] [NCP->driver mgmt ] [MGMT]
[01637208E185ACB16E6C0AE810636C69656E742F7365727665720000008FDE185ACB16E0000DBBC1A
081EAD1EFEFF9FFD90085B509BE6A]
[2018-12-05 13:41:20.568 + 0.001] [driver->app ] [MGMT]
[01637208E185ACB16E6C0AE810636C69656E742F7365727665720000008FDE185ACB16E0000DBBC1A
081EAD1EFEFF9FFD90085B509BE6A7EC18]
[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]
[60001641001F110AFDE185ACB16E00002F3683F40774DF19FF330040FDE185ACB16E000000000001163
31633001F542F5202544CB8]
[2018-12-05 13:43:36.668 + 0.008] [NCP->driver->IP stack ] [DATA]
[60001641001F110AFDE185ACB16E00002F3683F40774DF19FF330040FDE185ACB16E000000000001163
31633001F542F5202544CB8]

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]
[600A612D000C1140FDE185ACB16E00002F3683F40774DF19FDE185ACB16E00007434C3C136E3028C16
331633000C02D66000FB81]
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]
[600A612D000E1140FDE185ACB16E00002F3683F40774DF19FDE185ACB16E00007434C3C136E3028C16
331633000E197E6244FB83E7]

[2018-12-05 13:48:48.227 + 10.146] [NCP->driver->IP stack ] [DATA]
[60000000000211140FDE185ACB16E00007434C3C136E3028CFDE185ACB16E00002F3683F40774DF1916
3316330021F0634202FB84E7]
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]
[600A612D000E1140FDE185ACB16E00002F3683F40774DF19FDE185ACB16E00007434C3C136E3028C16
331633000E197C6244FB84E7]

[2018-12-05 13:48:58.406 + 10.146] [NCP->driver->IP stack ] [DATA]
[60000000000211140FDE185ACB16E00007434C3C136E3028CFDE185ACB16E00002F3683F40774DF1916
3316330021F0614202FB85E7]
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]
[600A612D000E1140FDE185ACB16E00002F3683F40774DF19FDE185ACB16E00007434C3C136E3028C16
331633000E197A6244FB85E7]

[2018-12-05 13:49:08.585 + 10.145] [NCP->driver->IP stack ] [DATA]
[60000000000211140FDE185ACB16E00007434C3C136E3028CFDE185ACB16E00002F3683F40774DF1916
3316330021F05F4202FB86E7]
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]
[600A612D000E1140FDE185ACB16E00002F3683F40774DF19FDE185ACB16E00007434C3C136E3028C16
331633000E19786244FB86E7]
```

```
[2018-12-05 13:49:18.765 + 10.146] [NCP->driver->IP stack ] [DATA]
[60000000000211140FDE185ACB16E00007434C3C136E3028CFDE185ACB16E00002F3683F40774DF1916
3316330021F05B4202FB88E7]
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]
[600A612D000E1140FDE185ACB16E00002F3683F40774DF19FDE185ACB16E00007434C3C136E3028C16
331633000E19746244FB88E7]

[2018-12-0513:49:28.945+10.146][NCP->driver->IP stack ] [DATA]
[60000000000211140FDE185ACB16E00007434C3C136E3028CFDE185ACB16E00002F3683F40774DF1916
3316330021F0594202FB89E7]
Received 43000 from client at fde1:85ac:b16e:0:7434:c3c1:36e3:28c
```

Client Host application:

For Client application, please use a Thunder Board 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.

3.22 QT Chemical Plant demo

3.22.1 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 are filling into mixture tank and stop when filling is done, and mixture tank is empty.

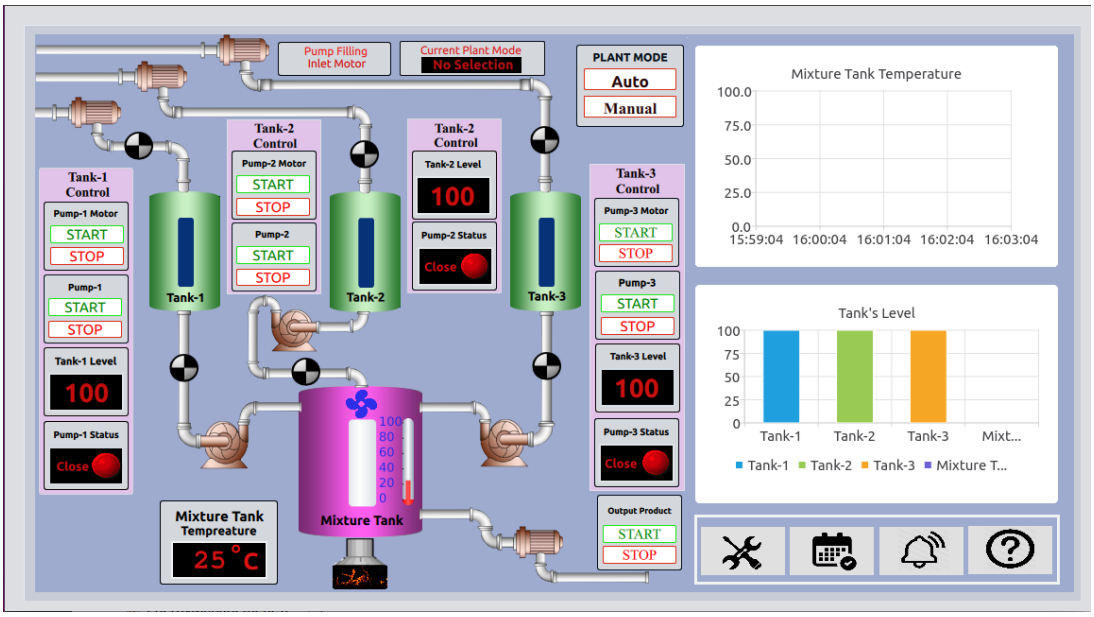

Figure 30 : QT Chemical Plant

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 the levels 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.

3.22.2 Steps to run Demo:

- Run command:

```
$ export DISPLAY=:0
$ QT_QPA_PLATFORM=xcb
$ /usr/share/chemicalplanthmi-1.0/ChemicalPlantHMI
```

- Click on “Auto” Plant mod
- 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.

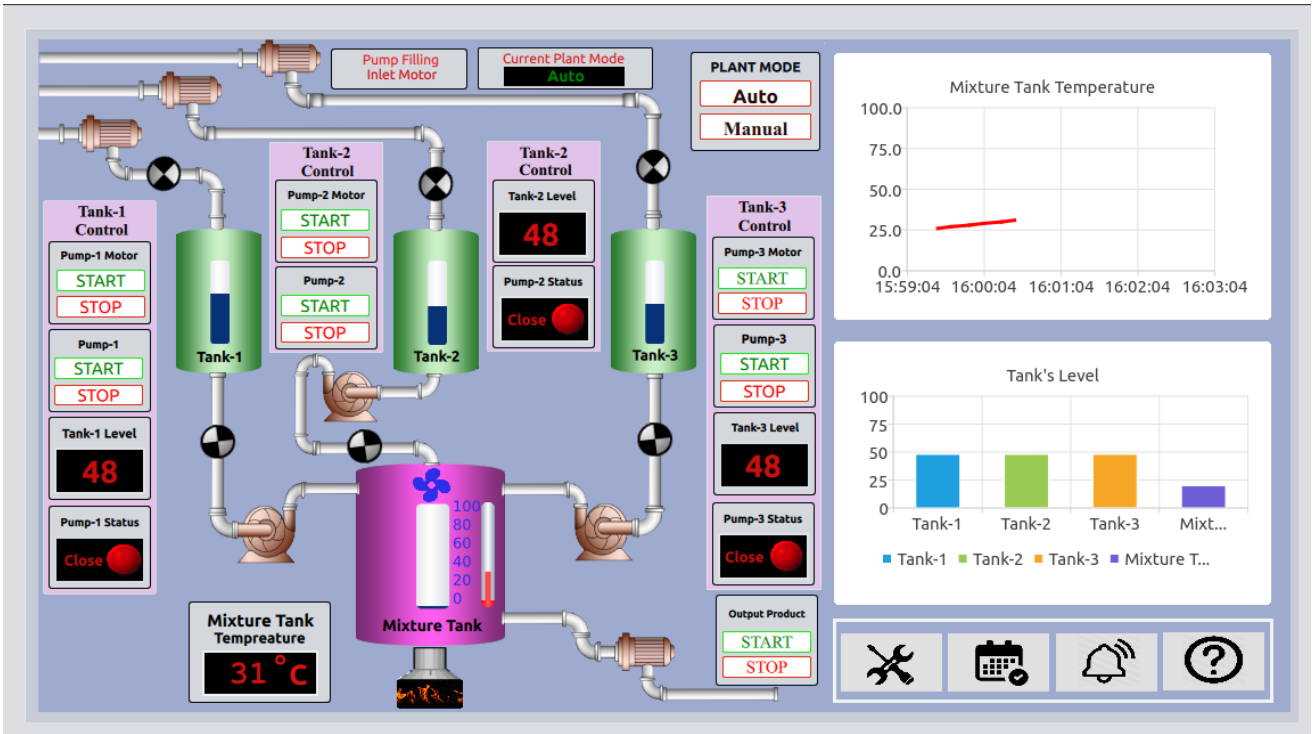

Figure 31 : Running Chemical Plant demo

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.

3.23 QT Video HMI Application demo

3.23.1 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)

3.23.2 Steps to run demo:

```
$ /usr/share/videohmiapp-1.0/VideoHMIApplication
```

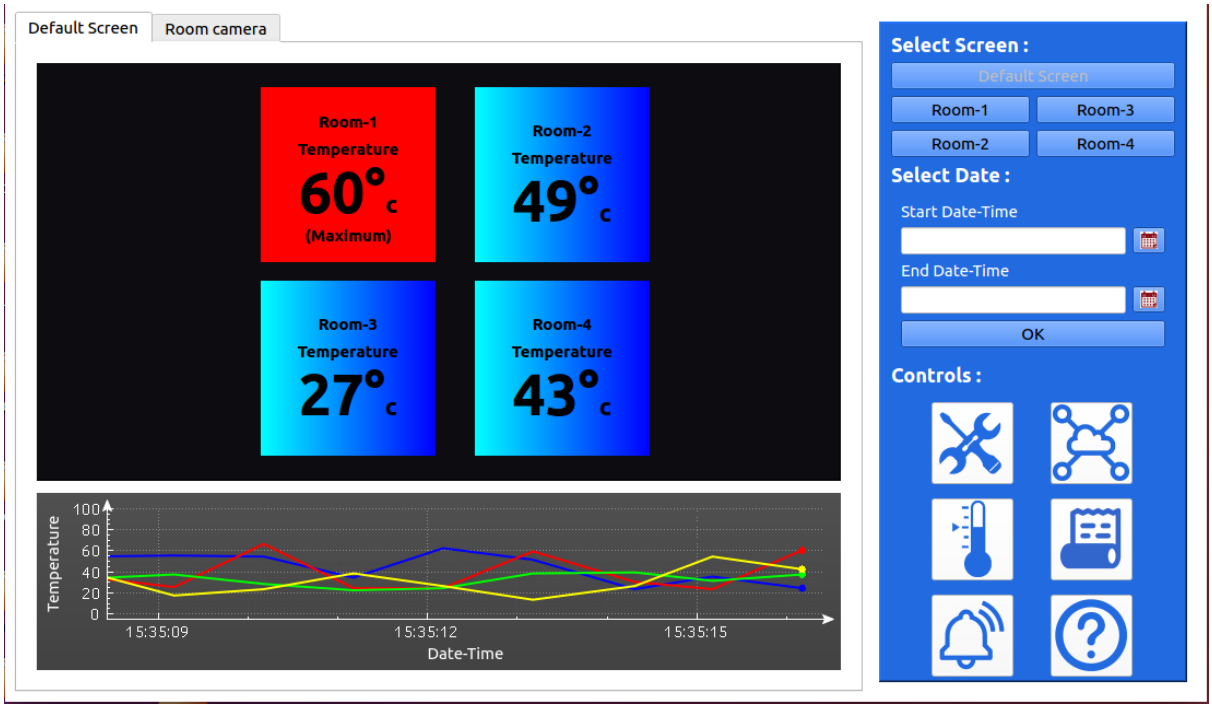

Figure 32 : HMI Video Application

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.

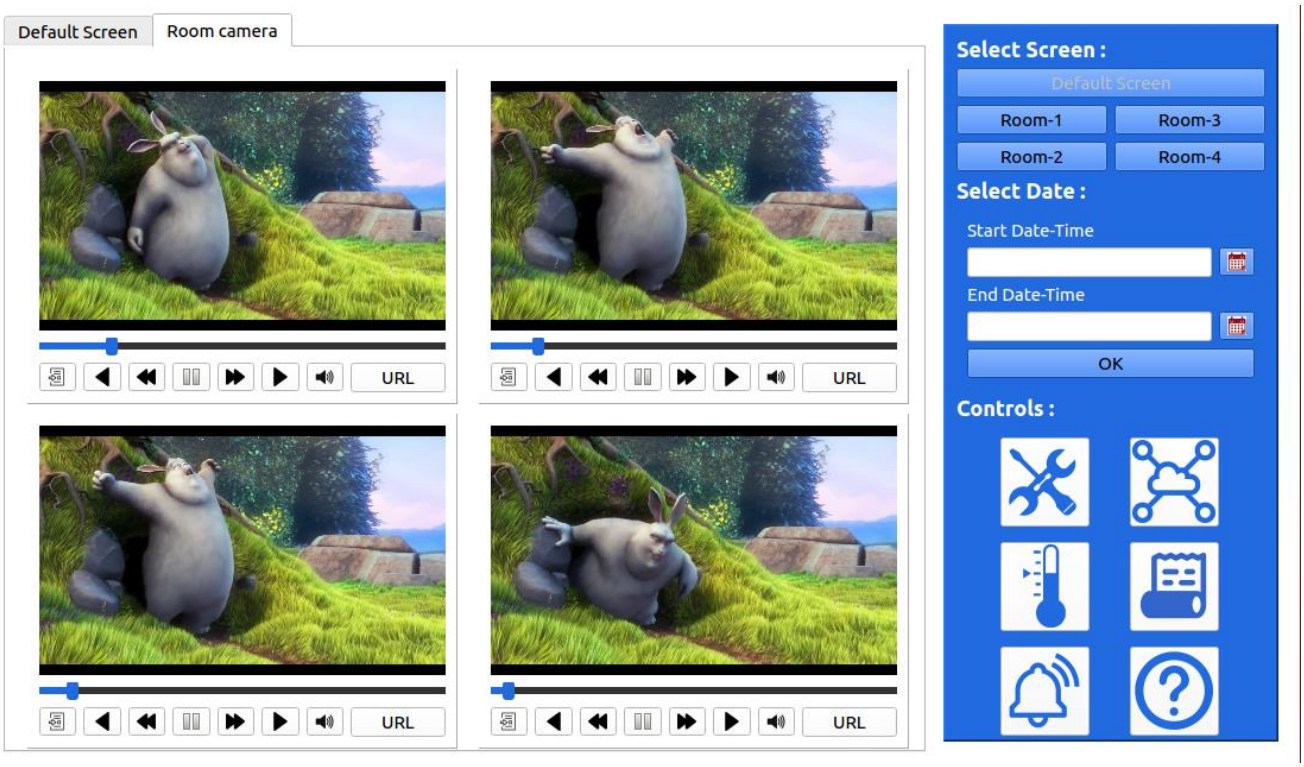

Figure 33 : Running multiple video files

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. Users need to give RTSP stream video command to live streaming. (Figure 20)

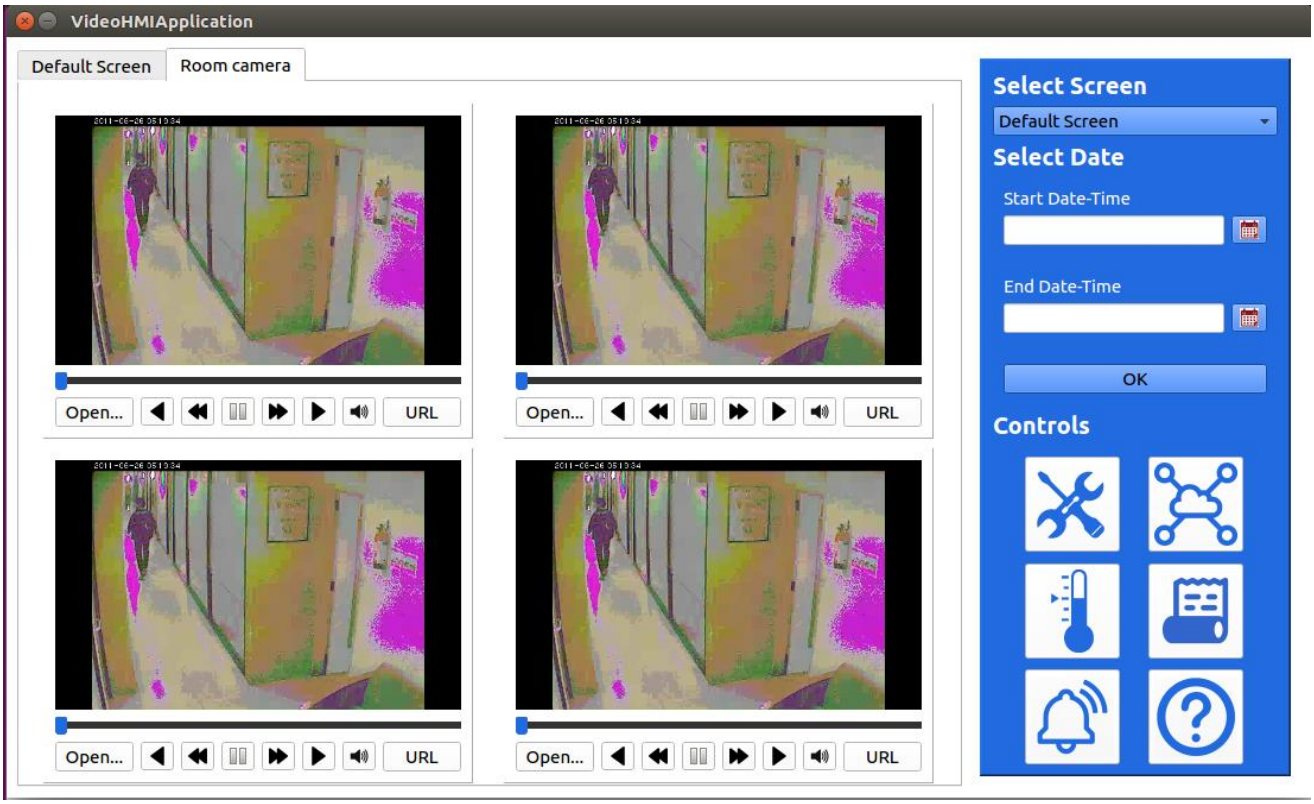

Figure 34 : Live Camera Streaming over RTSP

On Right – Top Corner, we have different Screen selection option. By default, default screen is selected, but user can go to camera screen to monitor that Room. For example, user can select Room-1 to monitor Room-1 condition. (Figure 34)

As shown in Figure 34, only Room-1 information is displayed. We can live stream or play video for room -1 only. Also, temperature graph shows 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 35)

As shown in Figure 36 based on selected date and time we show **Temperature History graph.**

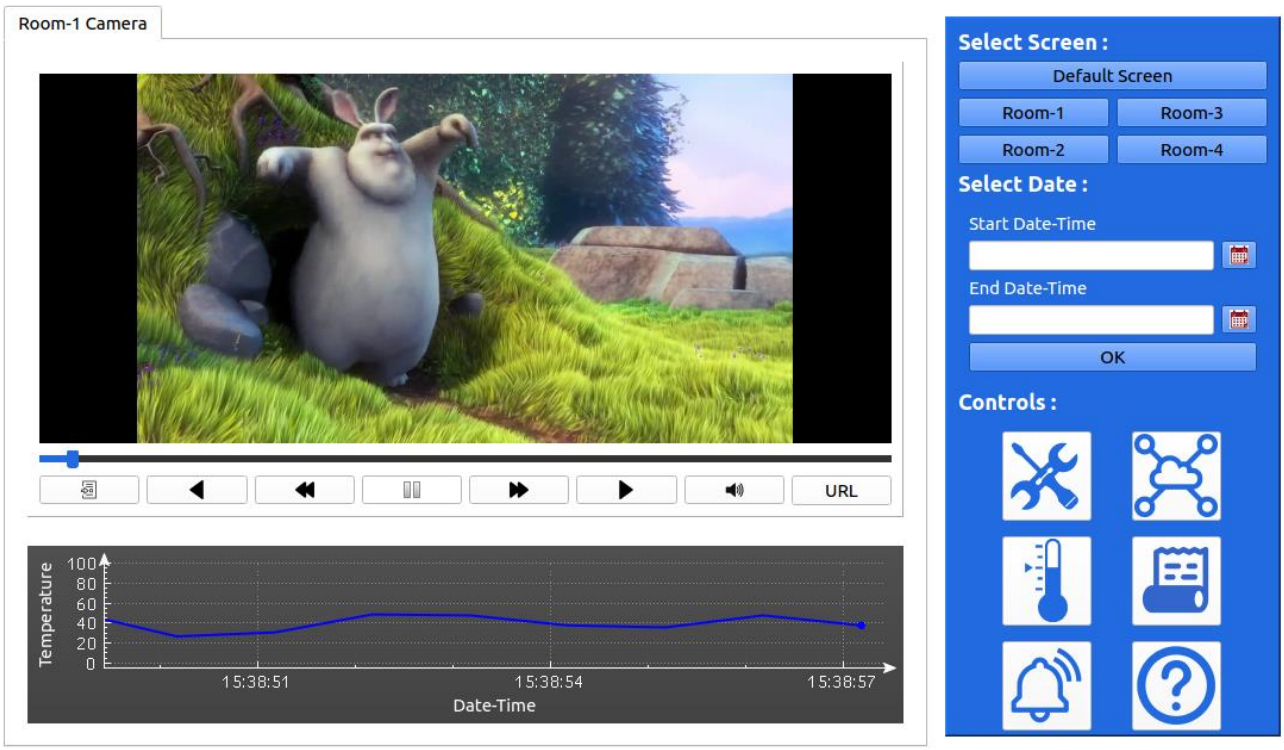

Figure 35 : Room-1 Screen

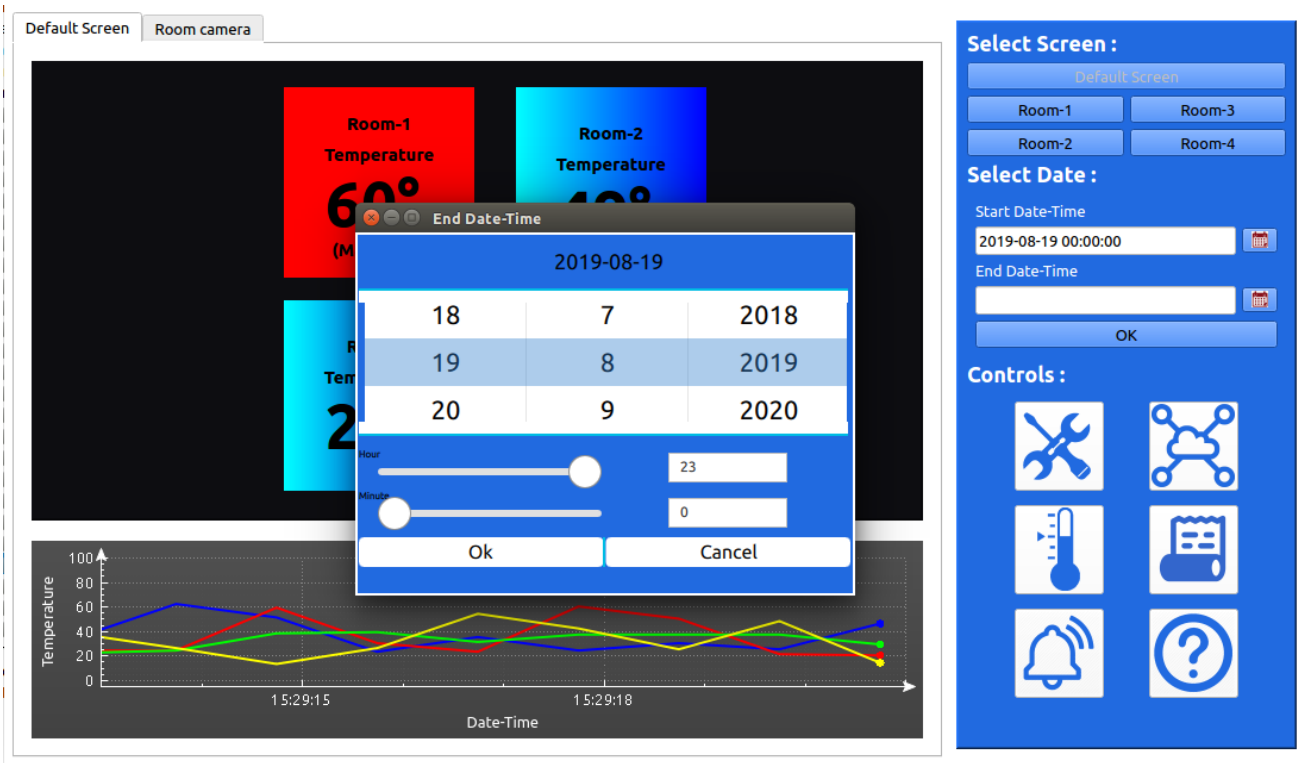

Figure 36 : Select DATE-TIME

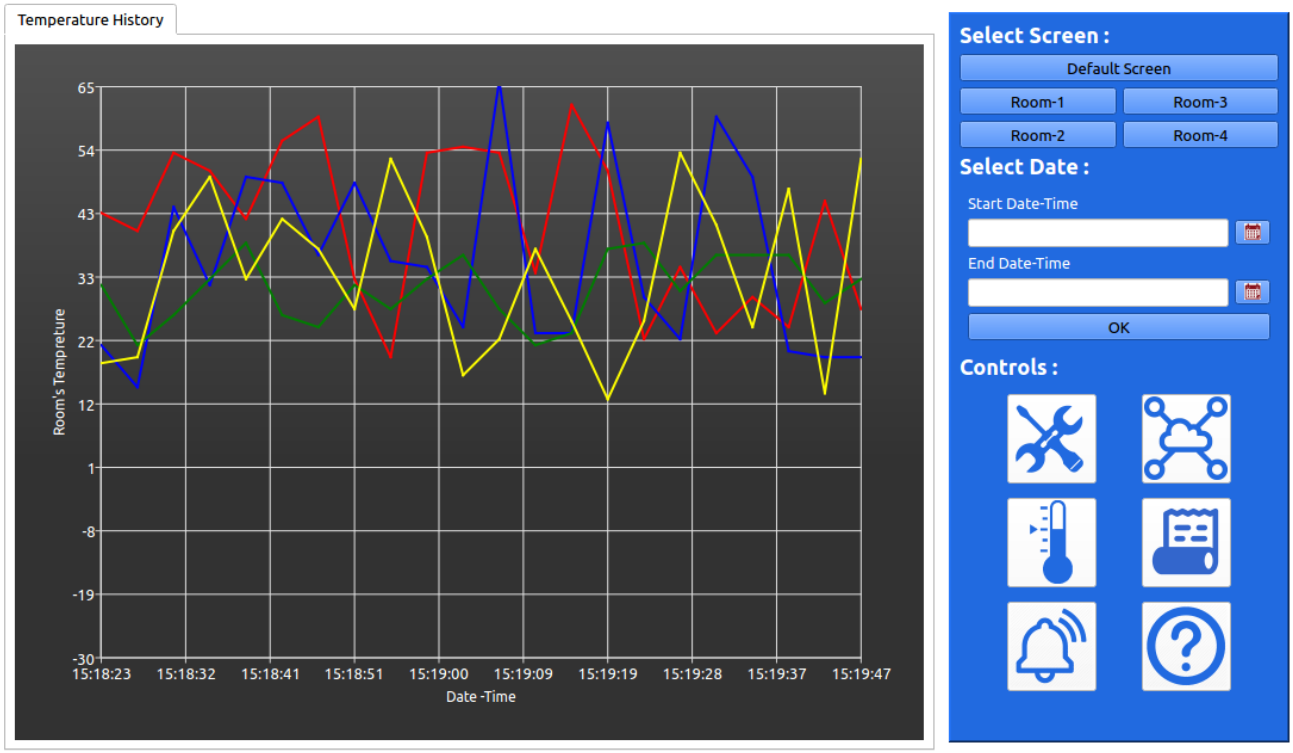

Figure 37 : Temperature History based on selected time

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

1. Settings:
   - Not implement
2. Data on cloud
   - Not implement
3. Temperature Setting: (Figure 35)
   - 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
5. Data History Table:
   - Display minimum or maximum temperature of the rooms with date and time

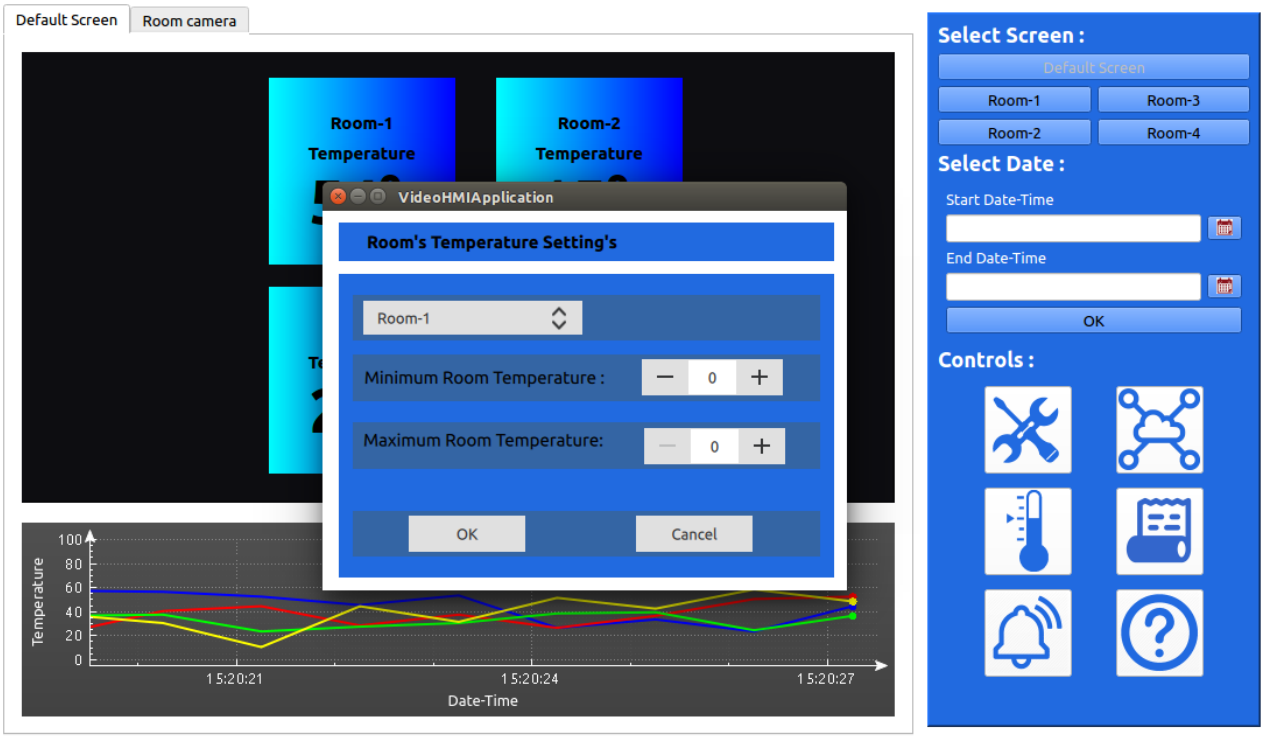

Figure 38 : Room Temperature Limit setting

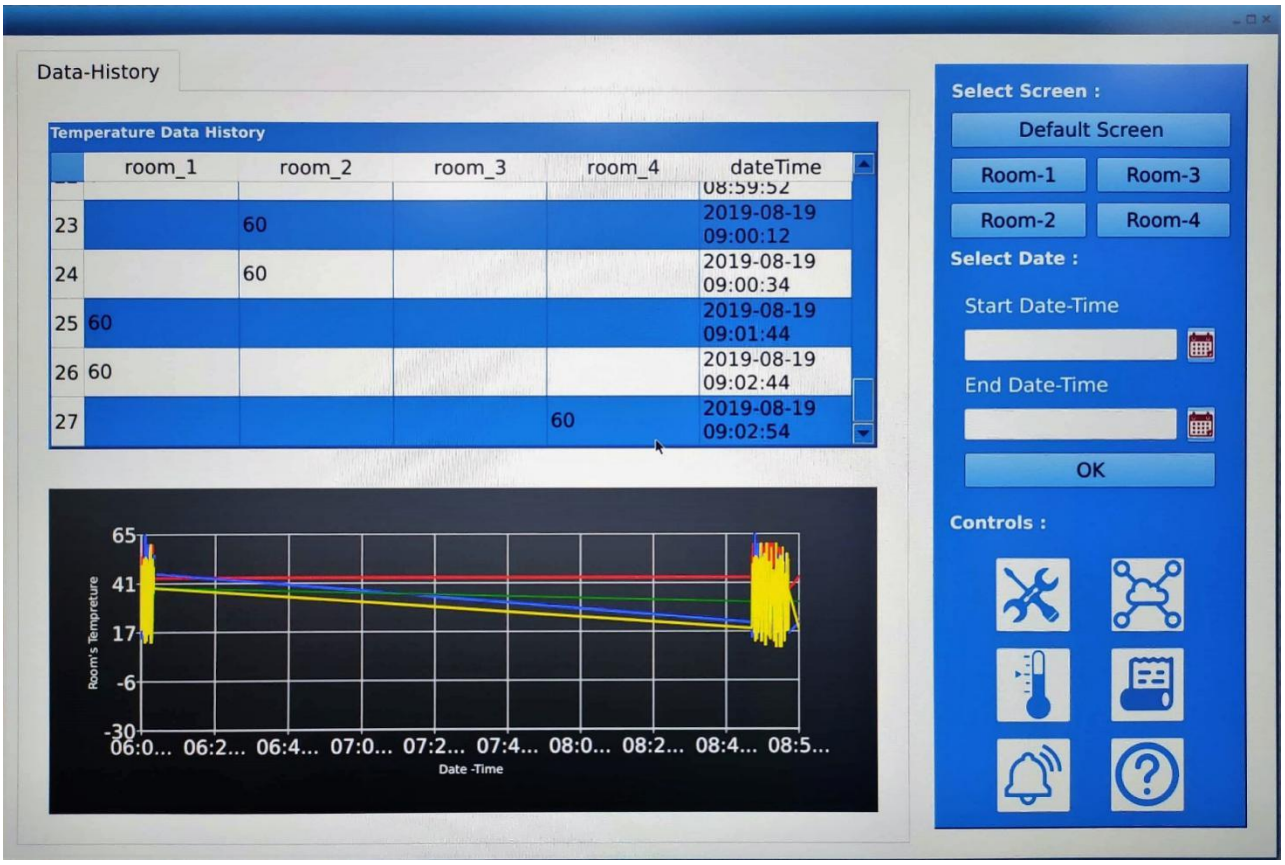

Figure 39 : Data History Screen

6. Data History graph:
    - Display history of all the room's temperature on the graph
7. Alarm
    - Not implement
8. 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.

3.24 Alexa Demo

We have prepared Alexa demo to showcase board's ZigBee capability. For that we have added support for ZIGBEE service, apache2 with cgi-bin and SSL, mosquito, php. Kindly go through **Alexa_User_Guide_vx.x user guide**.

3.25 ML and ARM NN demos

Please refer ML demo user guide, "**ML_Demos_Guide_iMX8_L5_10_Rel_4_0.pdf**" for this section.

4 LIMITATION

- Mezzanine DSI display is not available on 96boards.org so it is not validated

5 REFERENCES

[1] https://www.arrow.com/en/products/i.imx8-thor96/arrow-development-tools
[2] https://www.nxp.com/webapp/Download?colCode=L5.4.47_2.2.0_LINUX_DOCS
[3] https://www.nxp.com/docs/en/data-sheet/IMX8MDQLQIEC.pdf
[4] https://www.nxp.com/webapp/Download?colCode=IMX8MDQLQRM