MANIPAL ACADEMY of HIGHER EDUCATION
(Deemed to be University under Section 3 of the UGC Act, 1956)

MANIPAL SCHOOL OF INFORMATION SCIENCES
(A Constituent unit of MAHE, Manipal)

Firmware Development and Testing using QEMU

Reg. Number	Name	Branch
221039016	VARSHA V D	Embedded Systems
221039024	ARJUN M KRISHNA	Embedded Systems

Under the guidance of

Dr. Mohan Kumar J

Associate Professor,

Manipal School of Information Sciences,
MAHE, MANIPAL

10/11/2022

Contents

LIST OF FIGURES

ABBREVIATIONS

QEMU	Quick Emulator
KVM	Kernel-based Virtual Machine

ABSTRACT

QEMU is widely used in Cloud environments. Basically it is a fast and portable dynamic translator and an embedded machine emulator that emulates multiple CPUs and many board models. It can also provide a virtual platform for quickly software development such as Android Emulator, which uses it to emulate the whole mobile platform. But QEMU only supports common hardware, so new virtual device module should be developed for QEMU in order to emulate new hardware. Research result shows that the whole running, testing and debugging environment are built, and user level applications can be developed for the virtual hardware without the physical device become available.

Chapter 1

INTRODUCTION

Industrial Automation and Power technologies are characterized by extremely complex software systems which are in many cases very tightly coupled with mechanical and electrical sub-systems. These systems tend to be safety and mission critical. Development of such systems is considerably time consuming. However, they are also characterized by long if not very long product lifecycles sometimes in the order of fifteen years. The scope and complexity of software components in these industrial products has gone up significantly over the years. Continuous remodelling and development of software for the enduring industrial automation systems, places considerable challenges of efficiency on the development environment and processes followed. Here we look at the acceleration of embedded software development by decoupling it from the underlying hardware.

Traditionally, embedded software development is inherently dependent on hardware availability. Hardware development includes design, simulation and testing of hardware architecture, logic, circuit schematics and finally the PCB. This is a very time consuming process, often iterative and unpredictable due to dependency on factors like component availability and vendor support. Software developers usually cannot afford to wait for the entire hardware design to be completed and the board to arrive on their desks before starting on application development. This would increase the product development time substantially which is unacceptable. Application development as well as verification(testing) should be well underway by the time the hardware is finalized and delivered to the software developers. This is where system emulation and virtual environments step in and save the day. With the various advanced hardware emulators and virtual machines available today, embedded software developers can get a head start on their applications. There are few emulation products. SkyEye is an embedded emulator and mainly for ARM processors. Bochs is a platform emulator but only focus on X86. QEMU is a fast, portable and dynamic binary translator(DBT) that supports a wide range of processors(X86, ARM, PowerPC, MIPS etc), it can also emulate the whole platform not only PC but also embedded development board. In this work, we choose QEMU as target emulator.

1.1 Firmware

Firmware is a program on a hardware device that provides a control for a specific hardware device and gives instructions for how that hardware device communicates with other computer hardware. Firmware Development is the process of writing code that runs one embedded hardware rather than on a full-fledged computer. It is done on a microcontroller or a microprocessor. Firmware is not entirely considered to be software, but code that manages the sensors, peripherals, motors, and timers in devices.

Embedded firmware is responsible for controlling various peripherals of the embedded hardware and generating responses in accordance with the functional requirements for the particular product. Firmware is considered as the master brain of the embedded systems. Imparting intelligence to an embedded system is a one-time process and it can happen at any stage of the design. Once the intelligence is imparted to the embedded product, by embedding the firmware in the hardware, the product start functioning properly and will continue serving the assigned task till hardware breakdown occurs or a corruption in embedded firmware occurs. Designing an embedded firmware requires understanding of embedded product hardware like, various component interfacing, memory map details I/O port details, configuration and register details of various hardware chips used and some programming language.

1.2 QEMU

QEMU is a general-purpose, open-source machine emulator and virtualizer that can run operating systems and applications designed for one machine (such as an ARM board) on another when used as a machine emulator (e.g. your own PC). It runs on several host operating systems such as Linux, Windows and Mac OS X. The host and target CPUs can be different. QEMU is capable of emulating a complete machine in software without having actual hardware. It provides excellent system-level compatibility and support, making it ideal and lightweight virtual machine environment.

Chapter 2

LITERATURE SURVEY

In the beginning, the term emulation was usually used for virtualization using hardware. However, recently it has also become common to use the term emulation in the software context. Emulation refers to the replication of functions of a system by another, so that the emulator acts similar to the emulated system. There are various types of virtualization, depending on the system being virtualized and levels of abstraction used to achieve it. Hardware emulation refers to the virtualization of the required hardware on the host machine.

QEMU and Bochs are examples of hardware emulators. Native or Full virtualization uses a hypervisor which acts as a mediator between the operating system and the underlying hardware. This method is faster than emulation, however the criteria is that the underlying hardware must be supported by operating system. KVM and VMware are examples of full virtualization solutions. Paravirtualization uses a hypervisor for shared access to the underlying hardware. This method offers the best performance. However, since the virtualization code is integrated into the operating system itself, the guest operating systems have to be modified for the hypervisor. Xen and User Mode Linux are examples of paravitualization.

Chapter 3

OBJECTIVES & SPECIFICATIONS

3.1 Objectives

- ➢ To learn and understand the working of QEMU emulator.
- ➢ To emulate the available Firmware (Router/Any boards) on the QEMU emulator.
- ➢ To create an application for the emulated firmware.

3.2 Specifications

- ➢ To accomplish this, the target runs on the latest LTS version of Ubuntu Operating System 22.04.1 LTS.

Chapter 4

PROPOSED BLOCK DIAGRAM

4.1 Block Diagram

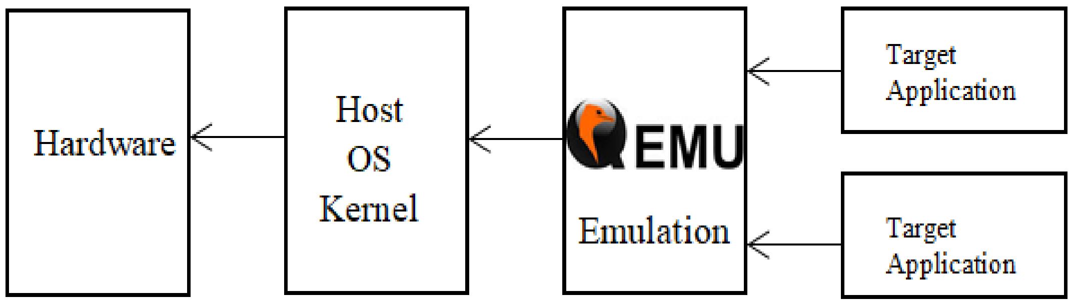


Fig 1: Proposed Blocked Diagram

When emulating there are some basic terms that we refer to the systems involved in the process. The two main components are the host machine and the target machine. The target is the system that we wish to emulate. The host is the machine that is used to build the target software. Here the host machine is Linux kernel and the target machine is Raspbian OS.

The Raspbian OS needs to be setup and emulated on QEMU using certain set of linux commands as shown below.

Ubuntu will be used to imitate the desired ARM versions. Get the newest Ubuntu version. You will require the following for the QEMU simulation:

- Raspbian OS image file.
- The corresponding latest qemu kernel for the Raspbian OS

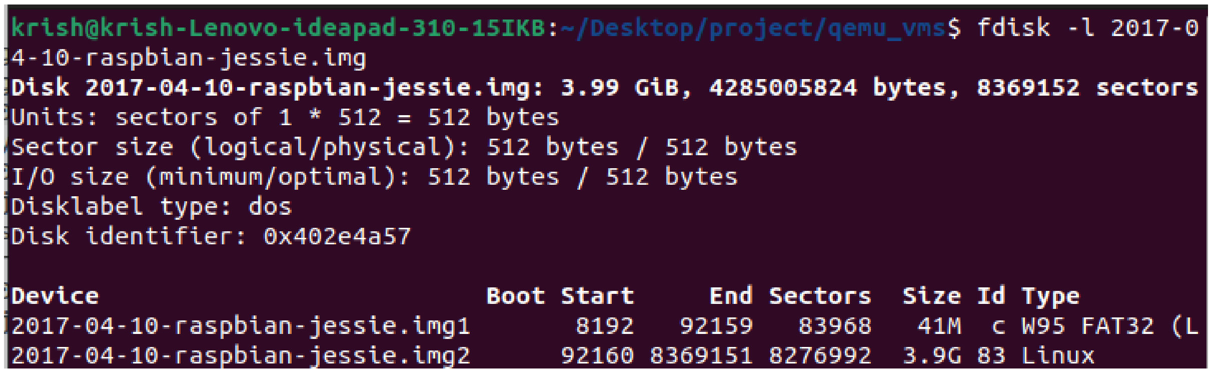
Make a new folder with the raspbian OS image file and the corresponding kernel and we can emulate raspbian on QEMU as follows.

Chapter 5

RESULT ANALYSIS

5.1 Results

1) fdisk is used to prepare and partition a brand new hardware. fdisk -l lists out all the options available for fdisk. We need the boot start address of 2017-03-02-raspbian-jessie.img2 and that corresponds to 92160 as shown in fig 2.

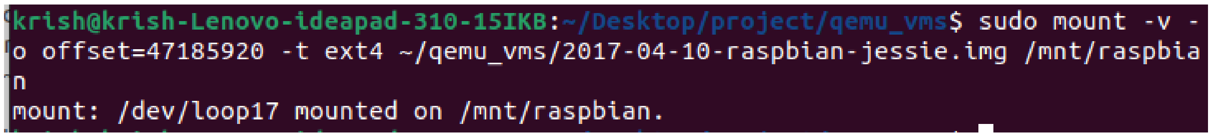


```
krish@krish-Lenovo-ideapad-310-15IKB:~/Desktop/project/qemu_vms$ fdisk -l 2017-0
4-10-raspbian-jessie.img
Disk 2017-04-10-raspbian-jessie.img: 3.99 GiB, 4285005824 bytes, 8369152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x402e4a57

Device                          Boot Start     End Sectors  Size Id Type
2017-04-10-raspbian-jessie.img1       8192   92159   83968   41M  c W95 FAT32 (L
2017-04-10-raspbian-jessie.img2      92160 8369151 8276992  3.9G 83 Linux
```

Fig 2: Snap of file system displaying boot start value.

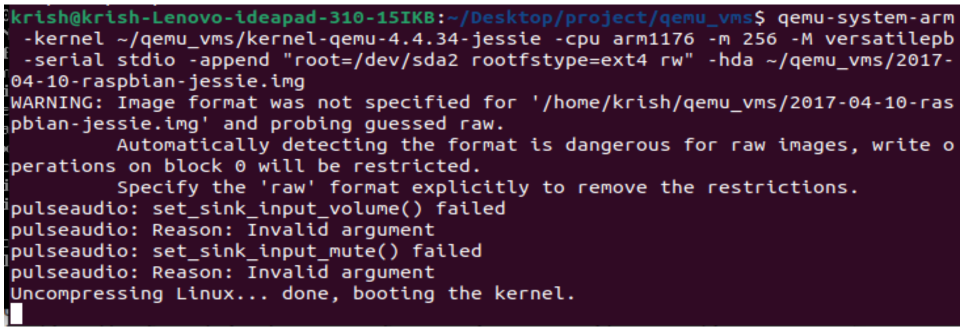
2) Next we need to mount the rasbian os image file with an offset which is equal to 512 x the boot start value obtained from above. That is 92160 x 512 = 47185920. We can mount the raspbian OS as shown in fig 3.



```
krish@krish-Lenovo-ideapad-310-15IKB:~/Desktop/project/qemu_vms$ sudo mount -v -
o offset=47185920 -t ext4 ~/qemu_vms/2017-04-10-raspbian-jessie.img /mnt/raspbia
n
mount: /dev/loop17 mounted on /mnt/raspbian.
```

Fig 3: Mounting Raspbian OS

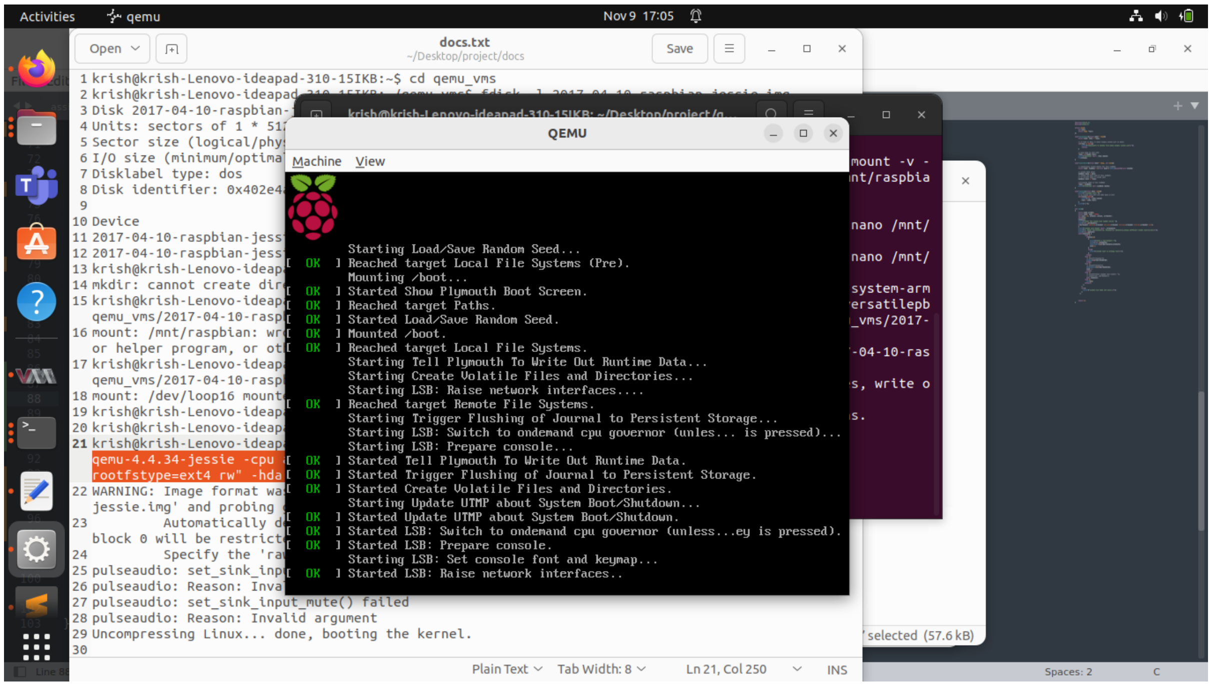
3) After mounting the raspbian image file, we will then emulate it using the “qemu-system-arm” command following with the kernel file – “kernel-qemu-4.4.34-jessie” along with specifying the Raspbian image file name as shown in fig 4.



```
krish@krish-Lenovo-ideapad-310-15IKB:~/Desktop/project/qemu_vms$ qemu-system-arm
 -kernel ~/qemu_vms/kernel-qemu-4.4.34-jessie -cpu arm1176 -m 256 -M versatilepb
 -serial stdio -append "root=/dev/sda2 rootfstype=ext4 rw" -hda ~/qemu_vms/2017-
04-10-raspbian-jessie.img
WARNING: Image format was not specified for '/home/krish/qemu_vms/2017-04-10-ras
pbian-jessie.img' and probing guessed raw.
         Automatically detecting the format is dangerous for raw images, write o
perations on block 0 will be restricted.
         Specify the 'raw' format explicitly to remove the restrictions.
pulseaudio: set_sink_input_volume() failed
pulseaudio: Reason: Invalid argument
pulseaudio: set_sink_input_mute() failed
pulseaudio: Reason: Invalid argument
Uncompressing Linux... done, booting the kernel.
```

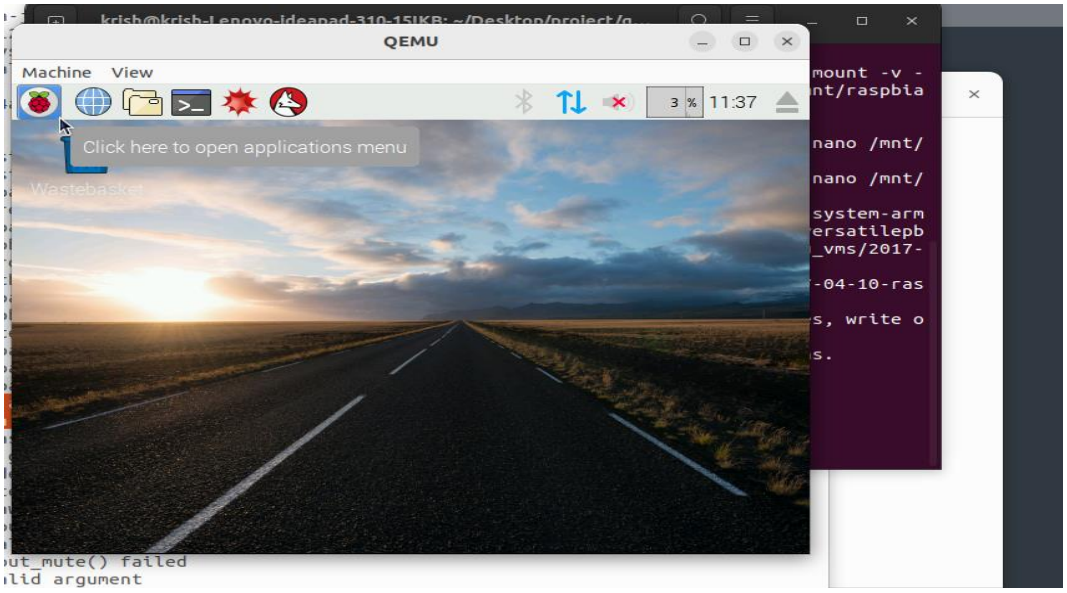
Fig 4: Emulation of mounted Raspbian OS

4) With these steps we will be able to emulate raspbian OS using QEMU. The below fig 5. shows the emulation of the Raspbian OS running on QEMU.



```
   Starting Load/Save Random Seed...
[  OK  ] Reached target Local File Systems (Pre).
         Mounting /boot...
[  OK  ] Started Show Plymouth Boot Screen.
[  OK  ] Reached target Paths.
[  OK  ] Started Load/Save Random Seed.
[  OK  ] Mounted /boot.
[  OK  ] Reached target Local File Systems.
         Starting Tell Plymouth To Write Out Runtime Data...
         Starting Create Volatile Files and Directories...
         Starting LSB: Raise network interfaces....
[  OK  ] Reached target Remote File Systems.
         Starting Trigger Flushing of Journal to Persistent Storage...
         Starting LSB: Switch to ondemand cpu governor (unles... is pressed)...
         Starting LSB: Prepare console...
[  OK  ] Started Tell Plymouth To Write Out Runtime Data.
[  OK  ] Started Trigger Flushing of Journal to Persistent Storage.
[  OK  ] Started Create Volatile Files and Directories.
         Starting Update UTMP about System Boot/Shutdown...
[  OK  ] Started Update UTMP about System Boot/Shutdown.
[  OK  ] Started LSB: Switch to ondemand cpu governor (unless...ey is pressed).
[  OK  ] Started LSB: Prepare console.
         Starting LSB: Set console font and keymap...
[  OK  ] Started LSB: Raise network interfaces..
```

Fig. 5: Emulation of the Raspbian OS running on QEMU.


krish@krish-Lenovo-ideapad-310-15IKB: ~/Desktop/project/q...
QEMU
Machine
View
Click here to open applications menu
Wastebasket
11:37

```
mount -v -
nt/raspbia

nano /mnt/

nano /mnt/

system-arm
ersatilepb
_vms/2017-

-04-10-ras

s, write o

s.
```

```
ut_mute() failed
lid argument
```

Fig.6: Emulated Raspbian OS

Advanced Networking

In some circumstances we want to have access to every port on the VM you are running in QEMU. As an illustration, let's say you run a programme that opens a network port or ports that you want to access or test from your host (Ubuntu) machine. We can construct a shared network interface (tap0) for this purpose, enabling us to access all open ports (if those ports are not bound to 127.0.0.1).

This can be done with the following commands on your HOST (Ubuntu) system:

```
$ sudo apt-get install uml-utilities
$ sudo tunctl -t tap0 -u <host-name>
$ sudo ifconfig tap0 172.16.0.1/24
```

After these commands you should see the tap0 interface in the ifconfig output.

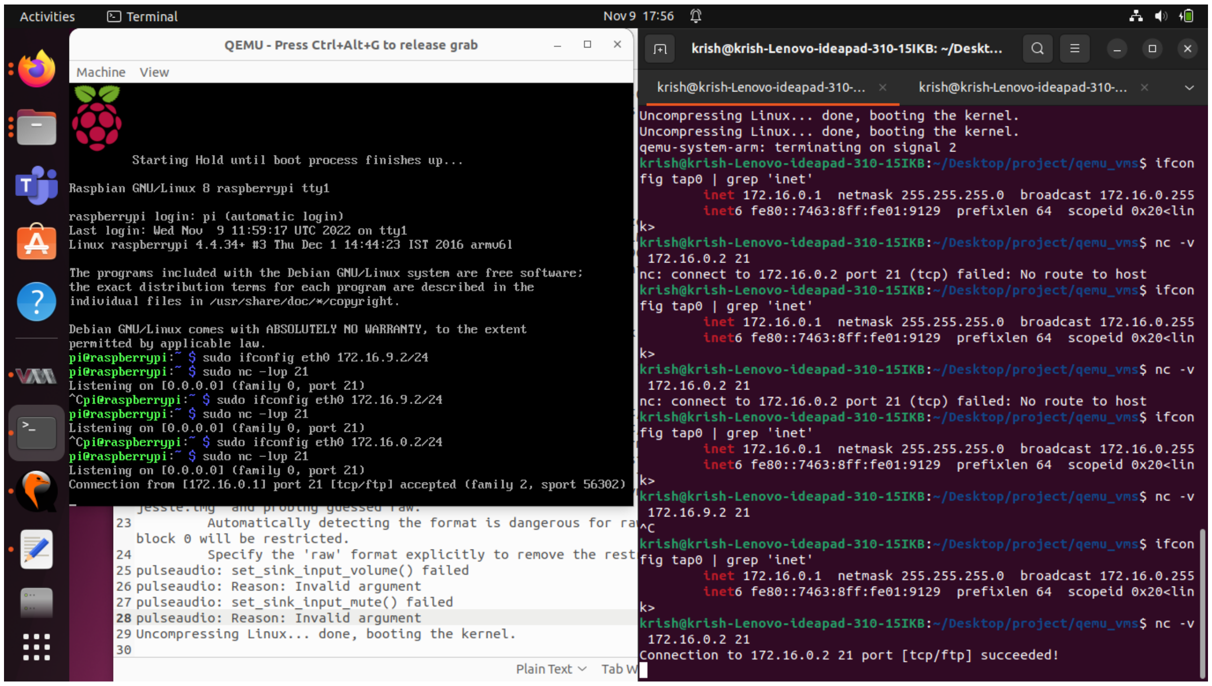
```
$ ifconfig tap0
```

You can now start your QEMU VM

When the QEMU VM starts, you need to assign an IP to its eth0 interface with the following command:

$ sudo ifconfig eth0 172.16.0.2/24

We were able to reach open ports on the GUEST (Raspbian) from your HOST (Ubuntu) system as shown in the fig 7.


Activities
Terminal
Nov 9 17:56
QEMU - Press Ctrl+Alt+G to release grab
Machine
View
krish@krish-Lenovo-ideapad-310-15IKB: ~/Deskt...
krish@krish-Lenovo-ideapad-310-...
krish@krish-Lenovo-ideapad-310-...
Plain Text
Tab W

```
        Starting Hold until boot process finishes up...

Raspbian GNU/Linux 8 raspberrypi tty1

raspberrypi login: pi (automatic login)
Last login: Wed Nov  9 11:59:17 UTC 2022 on tty1
Linux raspberrypi 4.4.34+ #3 Thu Dec 1 14:44:23 IST 2016 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
pi@raspberrypi:~ $ sudo ifconfig eth0 172.16.9.2/24
pi@raspberrypi:~ $ sudo nc -lvp 21
Listening on [0.0.0.0] (family 0, port 21)
^Cpi@raspberrypi:~ $ sudo ifconfig eth0 172.16.9.2/24
pi@raspberrypi:~ $ sudo nc -lvp 21
Listening on [0.0.0.0] (family 0, port 21)
^Cpi@raspberrypi:~ $ sudo ifconfig eth0 172.16.0.2/24
pi@raspberrypi:~ $ sudo nc -lvp 21
Listening on [0.0.0.0] (family 0, port 21)
Connection from [172.16.0.1] port 21 [tcp/ftp] accepted (family 2, sport 56302)
_
```

```
         Automatically detecting the format is dangerous for ra
block 0 will be restricted.
         Specify the 'raw' format explicitly to remove the rest
pulseaudio: set_sink_input_volume() failed
pulseaudio: Reason: Invalid argument
pulseaudio: set_sink_input_mute() failed
pulseaudio: Reason: Invalid argument
Uncompressing Linux... done, booting the kernel.
```

```
Uncompressing Linux... done, booting the kernel.
Uncompressing Linux... done, booting the kernel.
qemu-system-arm: terminating on signal 2
krish@krish-Lenovo-ideapad-310-15IKB:~/Desktop/project/qemu_vms$ ifcon
fig tap0 | grep 'inet'
        inet 172.16.0.1  netmask 255.255.255.0  broadcast 172.16.0.255
        inet6 fe80::7463:8ff:fe01:9129  prefixlen 64  scopeid 0x20<lin
k>
krish@krish-Lenovo-ideapad-310-15IKB:~/Desktop/project/qemu_vms$ nc -v
 172.16.0.2 21
nc: connect to 172.16.0.2 port 21 (tcp) failed: No route to host
krish@krish-Lenovo-ideapad-310-15IKB:~/Desktop/project/qemu_vms$ ifcon
fig tap0 | grep 'inet'
        inet 172.16.0.1  netmask 255.255.255.0  broadcast 172.16.0.255
        inet6 fe80::7463:8ff:fe01:9129  prefixlen 64  scopeid 0x20<lin
k>
krish@krish-Lenovo-ideapad-310-15IKB:~/Desktop/project/qemu_vms$ nc -v
 172.16.0.2 21
nc: connect to 172.16.0.2 port 21 (tcp) failed: No route to host
krish@krish-Lenovo-ideapad-310-15IKB:~/Desktop/project/qemu_vms$ ifcon
fig tap0 | grep 'inet'
        inet 172.16.0.1  netmask 255.255.255.0  broadcast 172.16.0.255
        inet6 fe80::7463:8ff:fe01:9129  prefixlen 64  scopeid 0x20<lin
k>
krish@krish-Lenovo-ideapad-310-15IKB:~/Desktop/project/qemu_vms$ nc -v
 172.16.9.2 21
^C
krish@krish-Lenovo-ideapad-310-15IKB:~/Desktop/project/qemu_vms$ ifcon
fig tap0 | grep 'inet'
        inet 172.16.0.1  netmask 255.255.255.0  broadcast 172.16.0.255
        inet6 fe80::7463:8ff:fe01:9129  prefixlen 64  scopeid 0x20<lin
k>
krish@krish-Lenovo-ideapad-310-15IKB:~/Desktop/project/qemu_vms$ nc -v
 172.16.0.2 21
Connection to 172.16.0.2 21 port [tcp/ftp] succeeded!
```

Fig 7: Established connection between HOST and target

5.2 Proposed work

Emulataion of Raspbian OS on QEMU:

Step 1: First, we installed the latest Ubuntu version and ran it on a VM

Step2: For the QEMU emulation we need the following:

- A Raspbian Image
- Latest qemu kernel

- Step 3: Inside the Ubuntu VM, we created a new folder and placed the Raspbian Jessie image and qemu-kernel into the folder created.

- Step 4: We emulated Raspbian OS on QEMU by following few linux commands.
  Here we see the GUI of Raspbian OS (Target application) from the Ubuntu System (Host).

Chapter 6

CONCLUSION AND FUTURE WORK

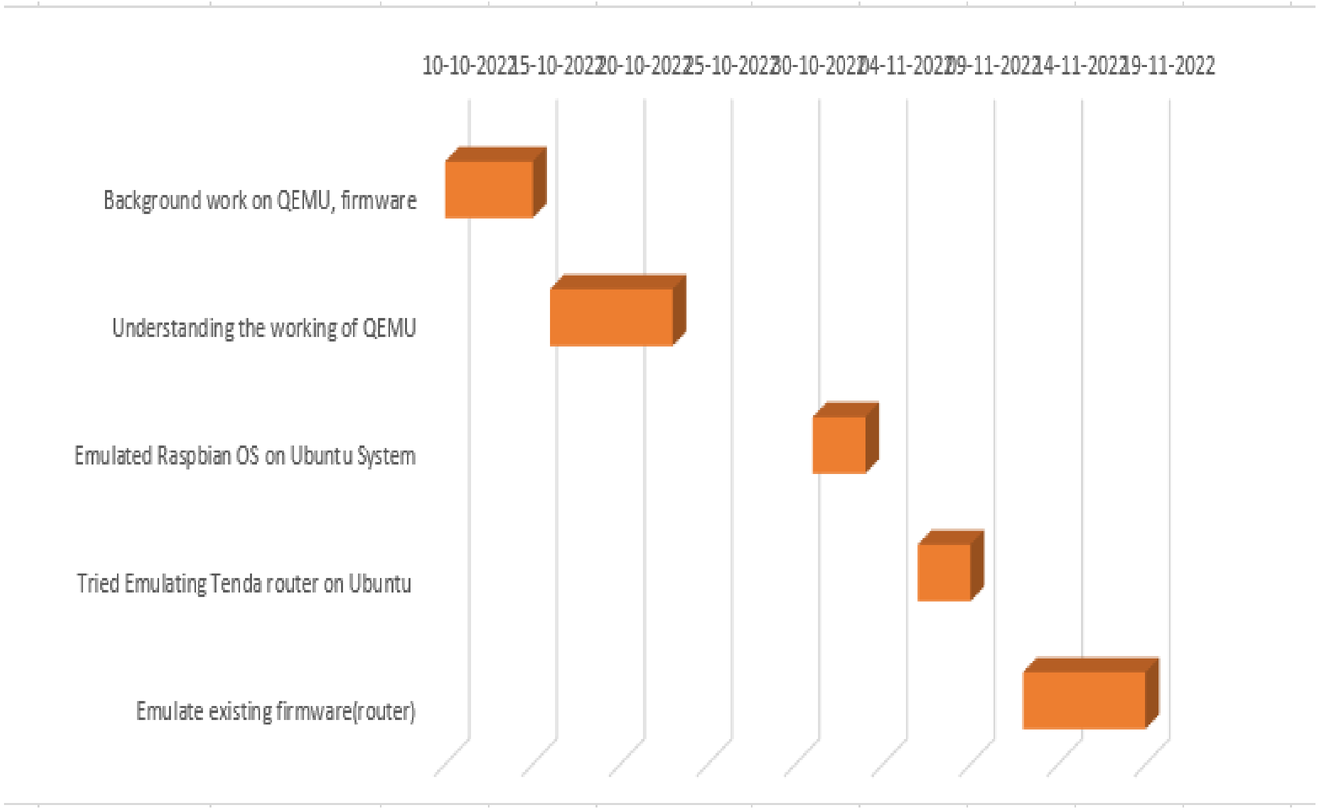
6.1 Conclusion

We emulated Raspbian OS environment on Ubuntu System without any access to the actual hardware.

6.2 Future work

For future work, we emulate the available firmware (router/any boards) on the QEMU emulator and create an application for the emulated firmware.

TIMELINE


10-10-2022
15-10-2022
20-10-2022
25-10-2022
30-10-2022
04-11-2022
09-11-2022
14-11-2022
19-11-2022
Background work on QEMU, firmware
Understanding the working of QEMU
Emulated Raspbian OS on Ubuntu System
Tried Emulating Tenda router on Ubuntu
Emulate existing firmware(router)

REFERENCES

[1] F. Bellard, "QEMU a Fast and Portable Dynamic Translator", *Proceedings of USENIX Annual Technical Conference*, pp. 41-46, June 2005.

[2] Y. Chen, J. Ren, H. Zhu and Y. C. Shi, "Dynamic binary translation and optimization in a whole-system emulator -SkyEye", *International Conference on Parallel Processing Workshops*, pp. 329-336, August 2006.

[3] XiaoXiao Bian , “Implement a Virtual Development Platform Based on QEMU”, *2017 International Conference on Green Informatics (ICGI).*

[4] *http://en.wikipedia.org/wiki/Virtualization*

[5] *https://azeria-labs.com*