Creating A Testbed for Flash Memory Research via LPC-H3131 and OpenNFM – Linux Version

Deepthi Tankasala, Niusen Chen, Bo Chen

Department of Computer Science, Michigan Technological University

bchen@mtu.edu

The Security and Privacy (SnP) Lab in Michigan Technological University has conducted extensive research on flash memory security [1-14], and most of our research relied on a flash memory testbed, which has been built using a cheap electronic development LPC-H3131 [15] and open-sourced flash controller OpenNFM [16]. A significant advantage of this testbed lies in its low cost, with less than $100 in total cost. Previously, we have published a technical report [17] which provides a guideline to researchers, educators and practitioners to set up this flash memory testbed when the host computer uses Windows. In this new technical report, we will provide a step-by-step guideline on how to set up this testbed when the host computer uses Linux.

The report outlines a list of required hardware and software in Sec. I, the necessary system configuration and software installation in Sec, II, and the detailed steps on how to cross-compile the open-source flash controller and flash it to the electronic development board in Sec. III.

I. The Required Hardware and Software

a. Hardware:

1. LPC-H3131 USB HEADER DEVELOPMENT PROTOTYPE BOARD [link]
2. USB A to B cable
3. USB A to Mini Cable
4. A host computer equipped with USB ports

b. Software:

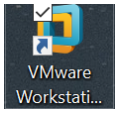
1. OpenNFM [link]: an open sourced flash memory controller
2. PuTTY [link]: a tool used to flash the flash memory controller (after compilation) to LPC-H3131. We used V4.105 in this technical report.
3. IAR Embedded workbench [link]: a cross compiler for OpenNFM. We used version 7.40.5.9739 in this technical report.
4. Microsoft Windows 10 operating system (for the host computer)
5. Virtual Machine Workstation 14 Pro[link]: version 14.1.8 build-14921873 is used to support Ubuntu operating system to run along with the host operating system and test the OpenNFM
6. Ubuntu ISO[link]: Ubuntu version 18.04.4-desktop-amd64 is used in VM to test the board.

II. System Configuration and Software Installation

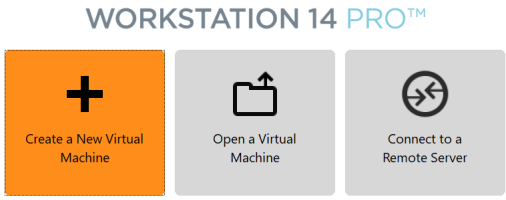
a. Ubuntu

To install PuTTY in Ubuntu operating system, we are using Virtual Machine Workstation Pro and VM installation is carried using the instructions in this link. Once the installation of VM is complete, Ubuntu ISO desktop-amd64 version 18.4 is installed using below steps:

1. After successful installation Virtual Machine Workstation, open the VM workstation by clicking on the icon that looks as below.

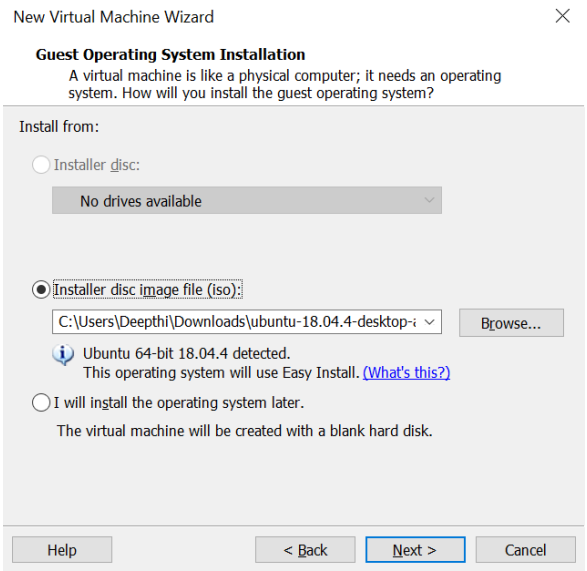


2. Home page of VM looks as below and select “Create a New Virtual Machine” option.

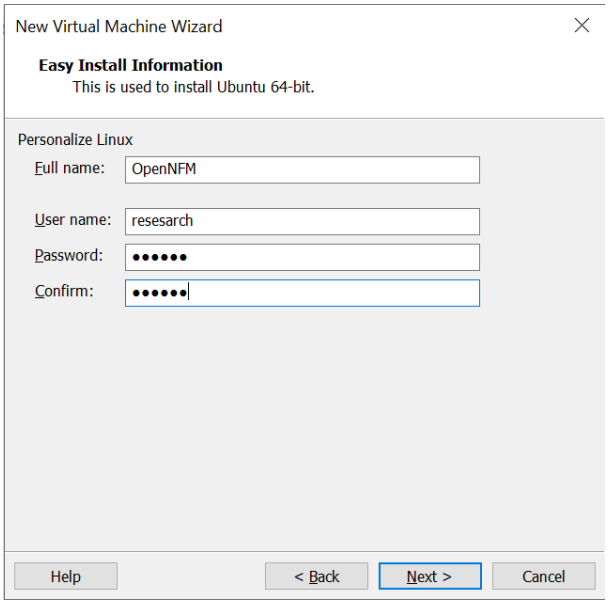


3. Virtual Machine Wizard is now open and select “Typical(recommended)” for configuration and click on Next.

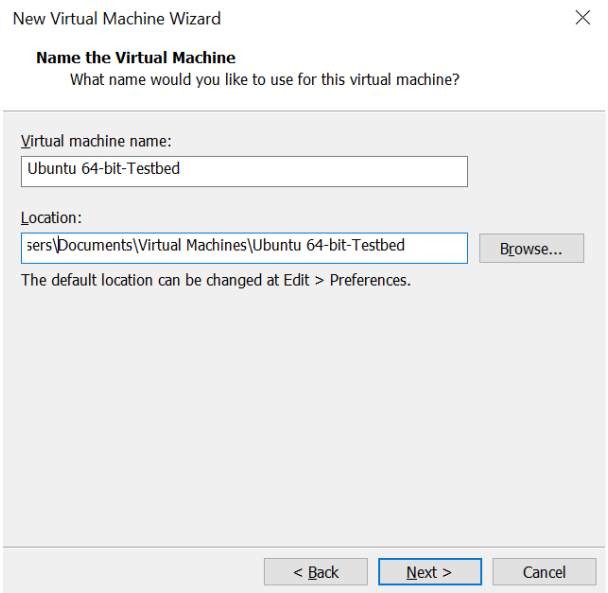
4. On the next screen, select “Installer disc Image (ISO)” and navigate to the location of Ubuntu 18.04.4-desktop-amd64.iso file and select the file as shown below and click next



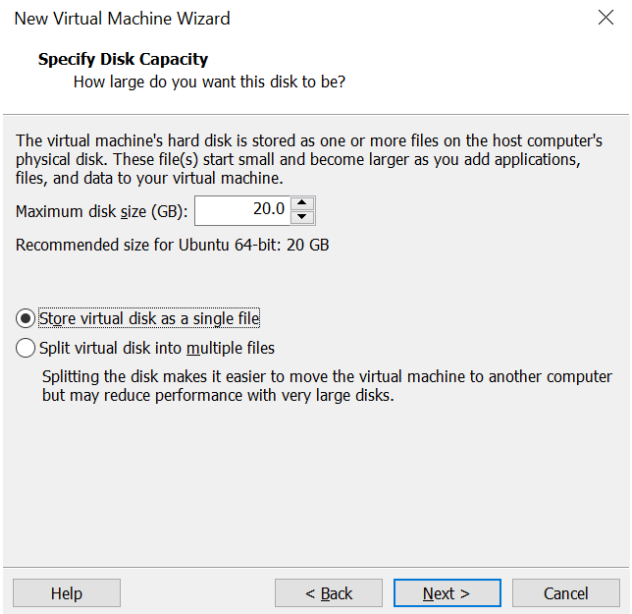
5. Enter Full name, User name and password fields in this screen as shown below and click on next



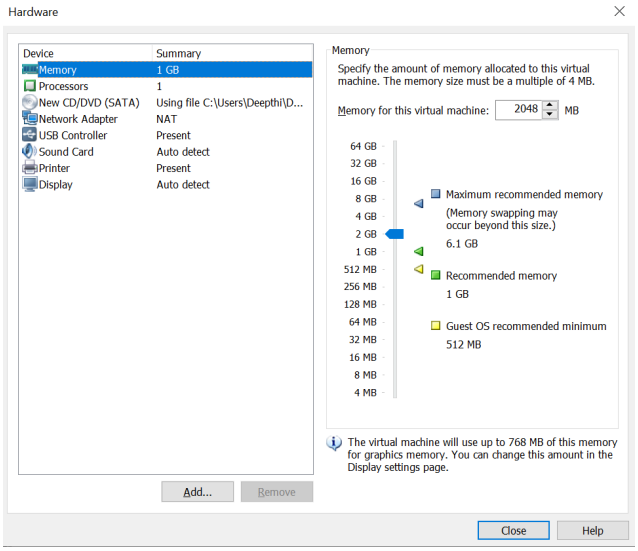
6. In the next screen, enter name and select the location for VM as shown below



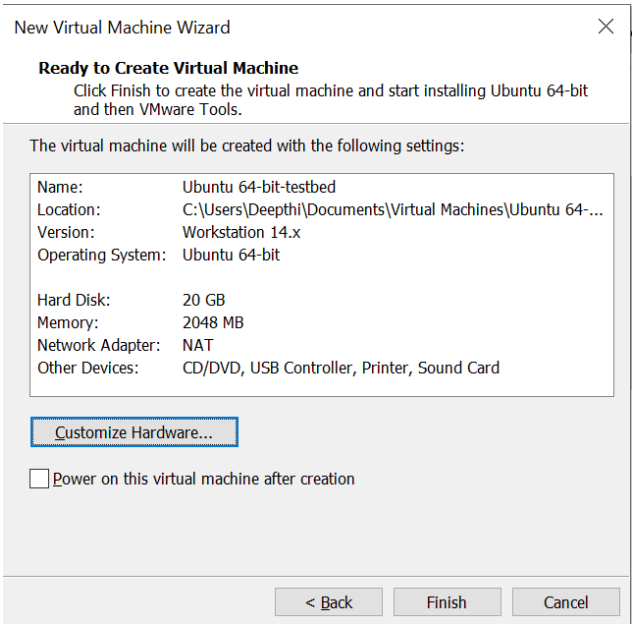
7. In the next screen, Maximum disc size, we select 20 GB and depending on the system capacity, users can change the capacity and select “Store virtual disc as single file” option and click on next as shown below



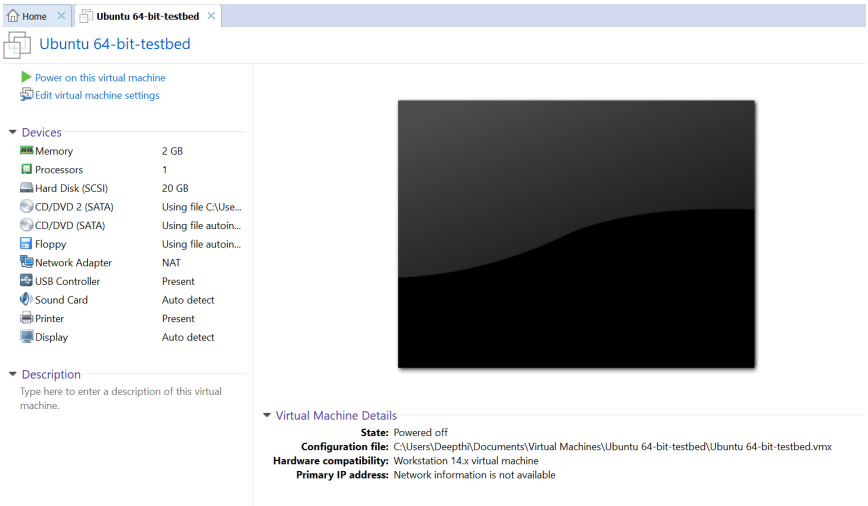
8. In the next screen, select customize hardware option and change the memory of VM to 2 GM or 2048 MB and click on close as shown below



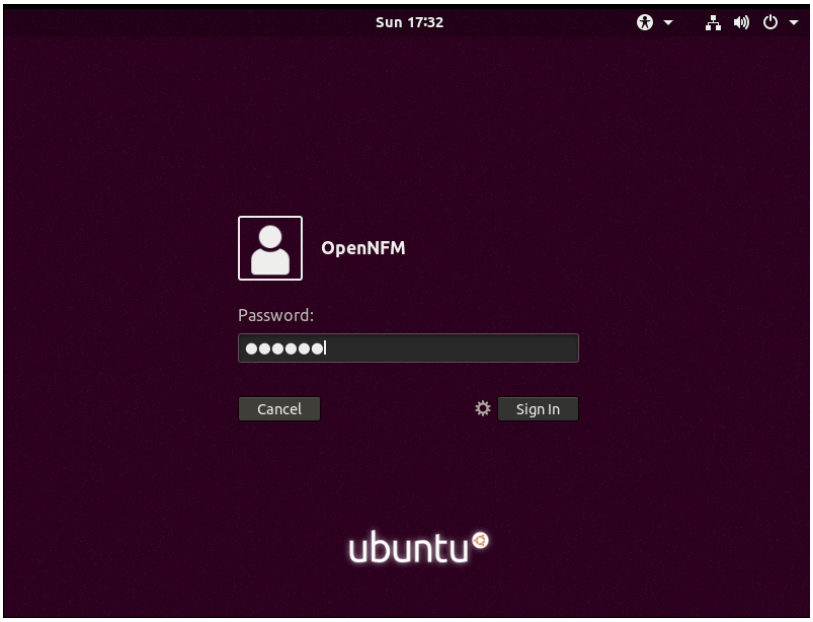
9. In the next screen, the summary is displayed, verify the details and click on Finish.



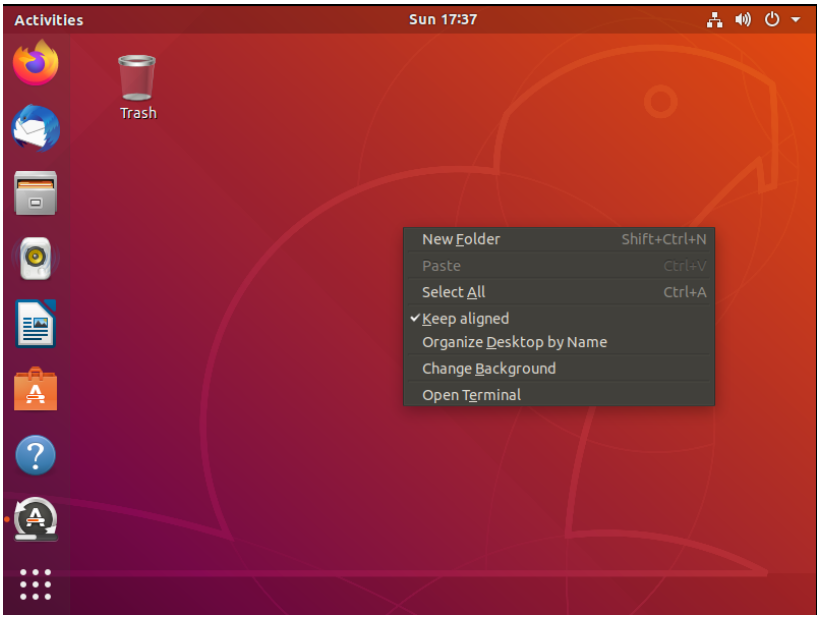
10. Home screen of VM is displayed as below and click on power on the virtual machine.



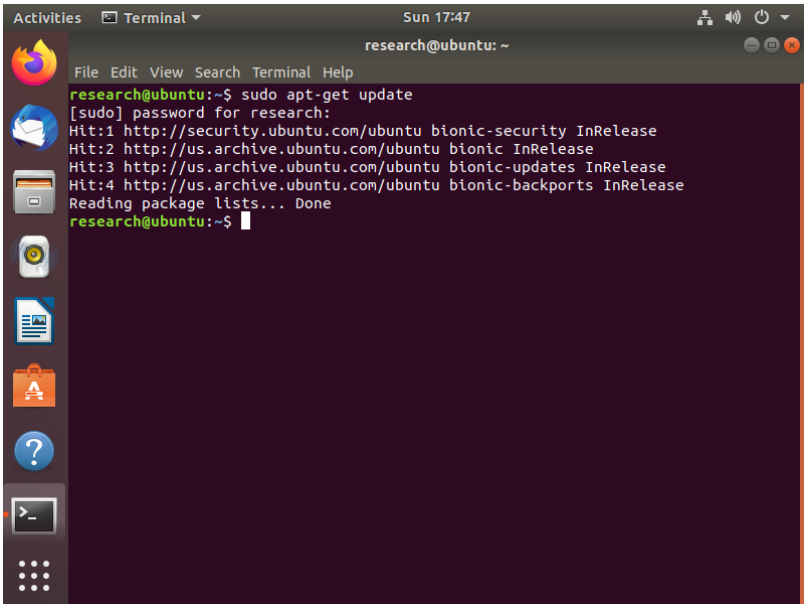
11. Once the installation is completed, login screen is displayed as below



12. On successful login, home screen of Ubuntu is displayed. Now right click on home screen to open the terminal as shown below



13. In the terminal use the command "**sudo apt-get update**" to check for updates and if any OS updates are present, system will be updating on executing this command above.

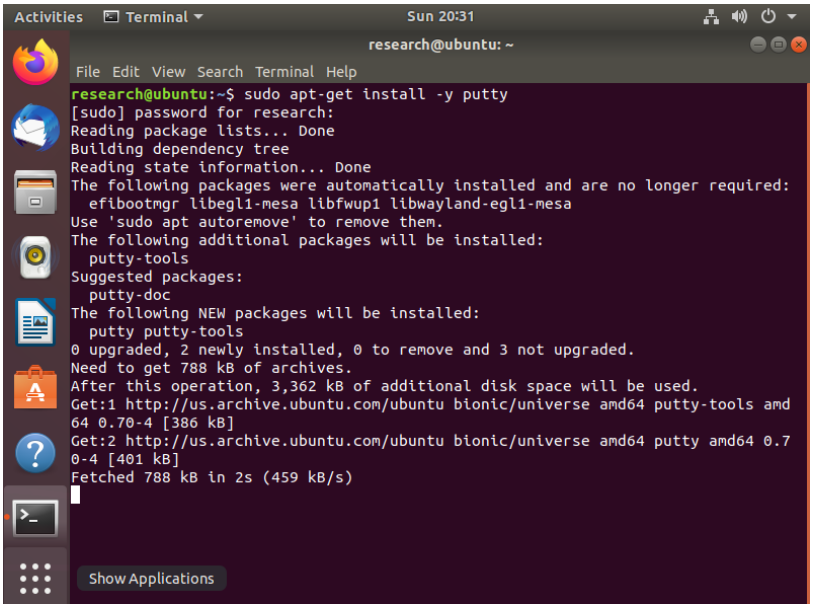


14. Ubuntu is now successfully installed.

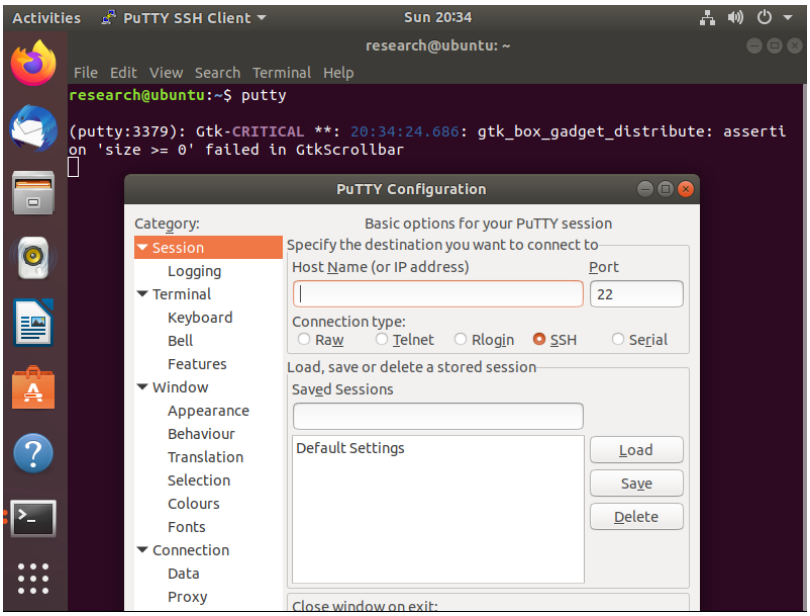
b. PuTTY

PuTTY is the application we are using for serial console and network file transfer. Below are the sets for setting up PuTTY on the Ubuntu VM installed in the above section II a.

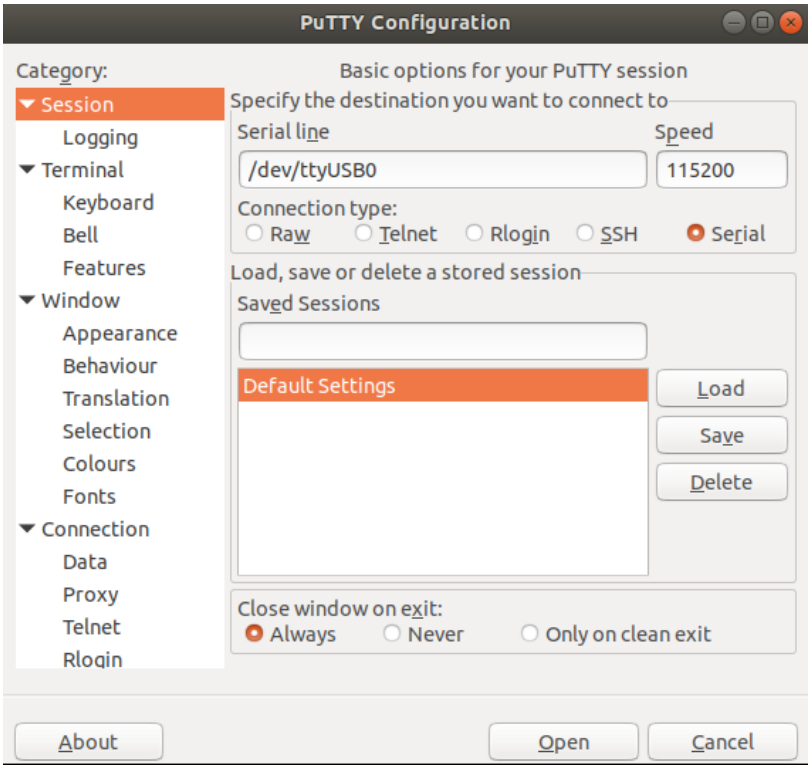
1. Firstly, connect the LPC board to the Desktop or the system you are using to create the testbed.
2. Open the terminal by right clicking and selecting Open Terminal option.
3. To install PuTTY, we need to run the terminal as Admin i.e. Root user. Use the command "**sudo apt-get install -y putty**" and run the command as shown below



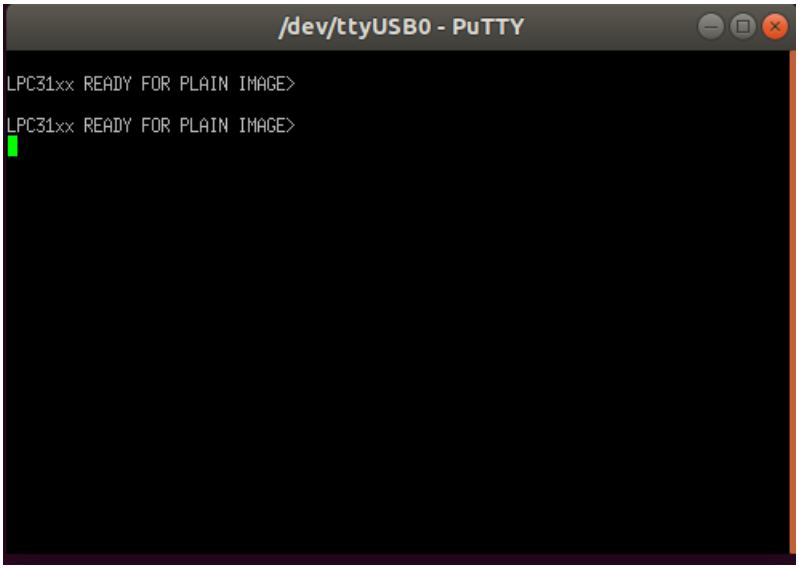
4. Once the PuTTY is successfully installed, open a new terminal and enter the command “putty” as shown in the below image. Putty configuration window should be now open.



5. Enter Host Name as “/dev/ttyUSB0”, select connection type as “serial”, change speed to “115200” and select “Default Settings” under saved sessions and click on save as shown in the below image. Saving the settings saves us time in future since we don’t have to set these configurations every time, we connect putty.



15. On successful connection, a new terminal will open and you can press the refresh button present on the LPC to check if the board is ready for Binary file transfer. A message “LPC21XX Ready for Plain Image” is displayed as shown below.



c. IAR Workbench

IAR workbench is used for executing the OpenNFM Code and generating the Binary file required for getting the LPC board started.

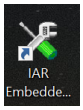
1. Free Version of IAR workbench is downloaded from this [link] and normal installation steps as directed by the installation is followed.
2. OpenNFM source code is added to the IAR Workbench and executed to generate “Binary File” that is used for setting up the test bed.

III. Compiling OpenNFM and Flashing The Binary to LPC-H3131

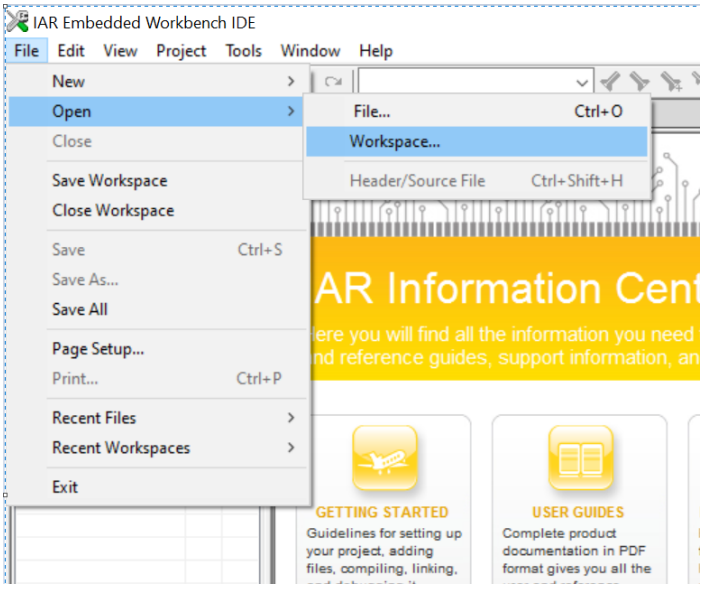
a. Cross-compiling OpenNFM

OpenNFM code base contains all the code required for creating our test-bed for NAND flash research. To compile OpenNFM, the project must be opened in IAR Workbench. Since we already have installed the IAR workbench, below are the steps for compiling and generating binary file required for our test bed.

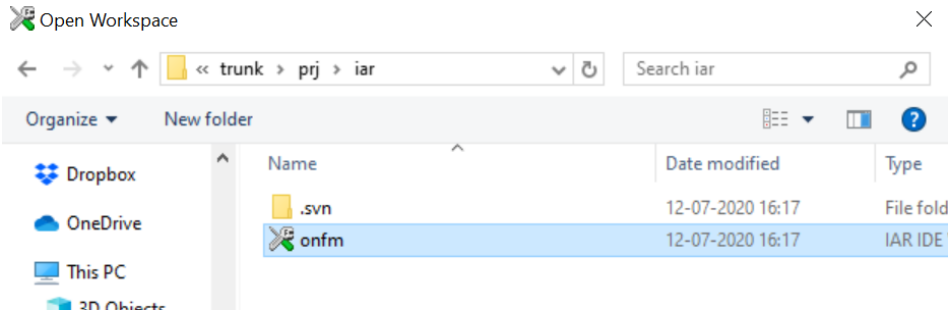
1. Open IAR Workbench by clicking on the application icon which looks like this.



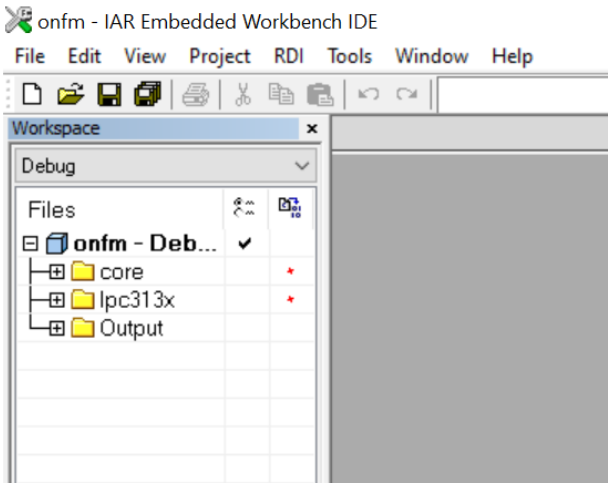
2. Once the application is open, go to menu bar and click the file option to find "open" and select "Workspace" option.



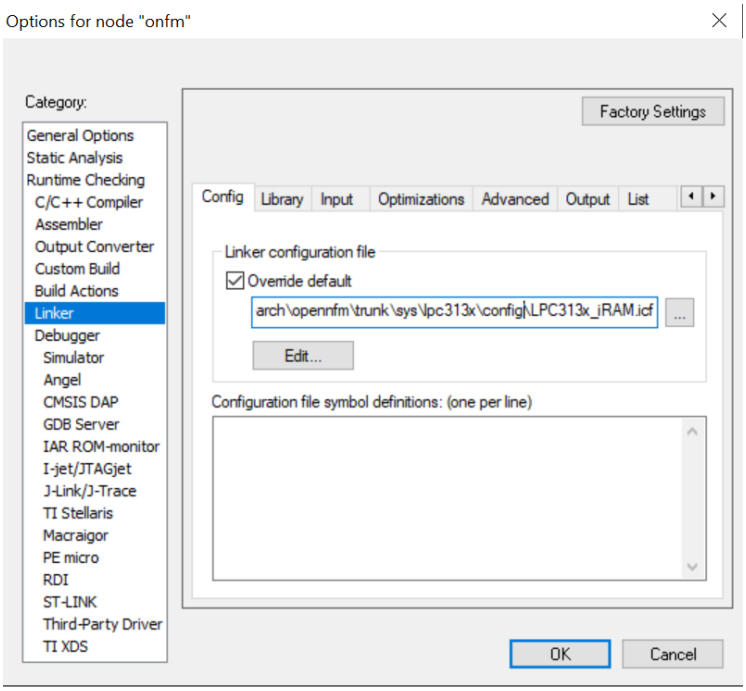
3. A window opens and we need to open IAR IDE Workspace type file from OpenNFM code. From the window, navigate to the OpenNFM code location and go to opennfm>>prj>>iar and double click on: "onfm" an IAR IDE workbench file.



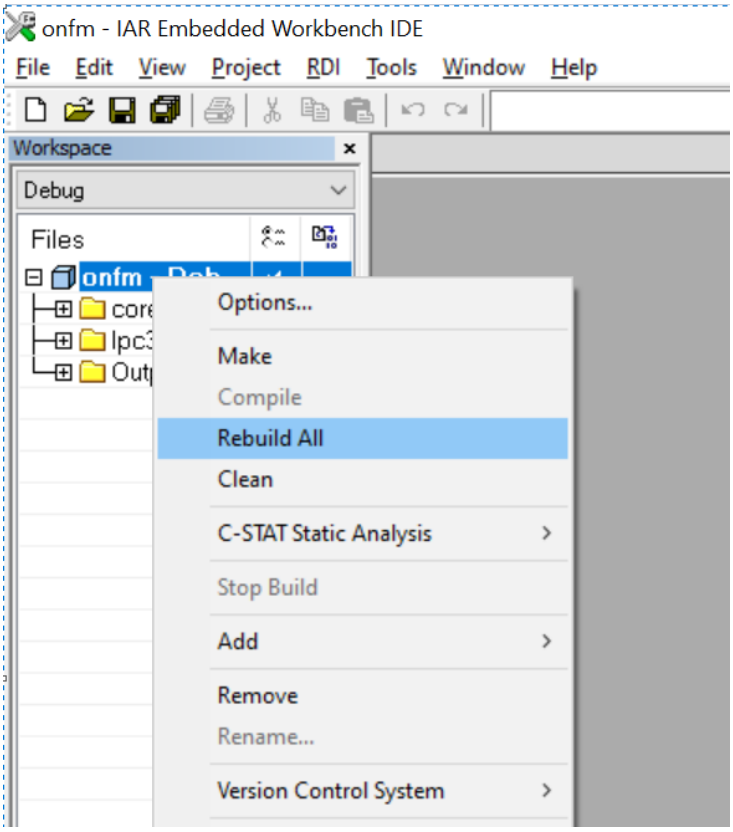
4. Now the OpenNFM is imported into IAR Workbench. And the all the workbench structure is displayed under Debug panel of IAR workbench as below.



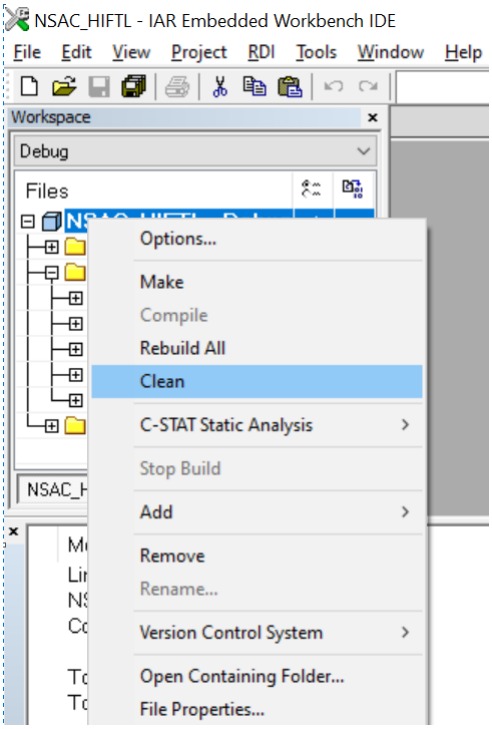
5. We need to set the linker settings for this project. Go to Project option on the menu bar and click on linker option and under override default option, add the address location of “LPC313x_iRAM.icf” file present in the OpenNFM code folder Opennfm>>trunk>>sys>>lpc313x>>config>> LPC313x_iRAM.icf and click ok.



6. To make sure, all the files are present, you need to check if the code compiles by right clicking on the project present under the “Debug” panel and select “Rebuild All” option.



7. If all the files are intact, the project should compile successfully without errors.
8. If any errors are present, the details of the error are displayed under the build panel of the IAR workbench.
9. To clear the error messages under the build panel and to rebuild the entire project, right click on the project and click on “clean” option



10. On successful compilation of the OpenNFM code, binary file with the name “onfm” will be generated in the Exe folder present in the below location

11. Opnfm>>prj>>iar>>Debug>>Exe
12. To verify whether the binary file is latest or not, check the Date modified property of the opennfm (OUT File type). The time should match the time when OpenNFM project was rebuild from IAR Workbench.
13.

log.txt	25-06-2020 13:25	Text Document	3 KB
onfm	13-07-2020 20:23	OUT File	420 KB
onfm_fresh.bin	13-07-2020 20:23	BIN File	23 KB

14. Whenever any changes are made to the files in the project folder, the entire code should be re-compiled and then newly generated “onfm” binary file should be uploaded in the Tera Term to see the updated changes in action on the OpenNFM test bed simulation.

b. Flashing The Binary of OpenNFM to LPC-H3131

For Ubuntu operating system, we are using PuTTY application for serial console and network file transfer. Below are the steps:

1. We will need below mentioned .sh files for starting the process
   a) download.sh:

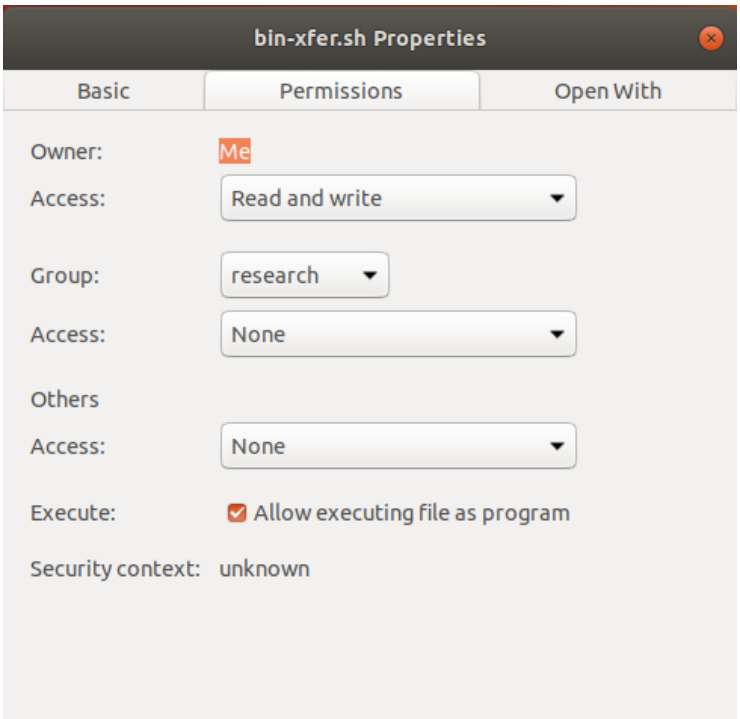
```
sudo ./bin-xfer.sh -i NSAC_DEBUG.bin -o /dev/ttyUSB0
```

   b) bin-xfer.sh:

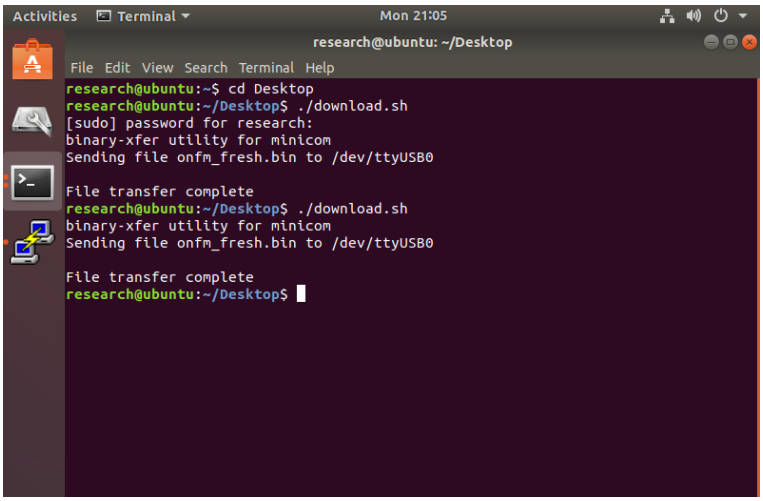
```
#!/bin/sh
INFILE=/dev/null
OUTFILE=/dev/null

exists() {
 command -v $1 >/dev/null 2>&1
}
while [ $# -gt 0 ]; do
 case "$1" in
 -i)
 shift
 INFILE="$1"
 ;;
 -o)
 shift
 OUTFILE="$1"
 ;;
 -h|--help)
 echo "$0 -i infile -o outfile"
 ;;
 *)
 INFILE="$1"
```

```
 esac
 shift
done
cat << EOF
binary-xfer utility for minicom
Sending file ${INFILE} to ${OUTFILE}
EOF
if (exists pv); then
  pv --force -i 0.25 -B 128  ${INFILE}  2>&1 > ${OUTFILE}
else
  cat ${INFILE} > ${OUTFILE}
fi
cat << EOF
File transfer complete
EOF
sleep 1
```

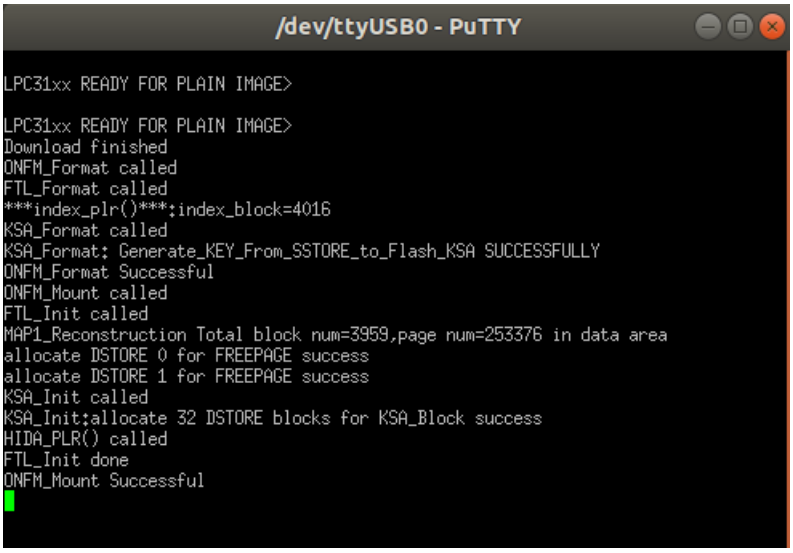
2. Onfm_fresh.bin file is taken from the location **Opnfm>>prj>>iar>>Debug>>Exe** folder after successfully rebuilding the workspace.
3. Copy all the three folders into Ubuntu VM currently running from Sec II (a)
4. If PuTTY is not currently running or disconnect, follow the steps in Sec II (b) to start the connection.
5. All the above 3 files are copied on to Ubuntu Desktop for convenience. User can choose any location/folder to copy these files accordingly.
6. Right click on bin-xfer.sh and go to properties and check “Execute: Allow executing file as program” and Access to “Read and Write” and repeat the same process for download.sh file as shown below.



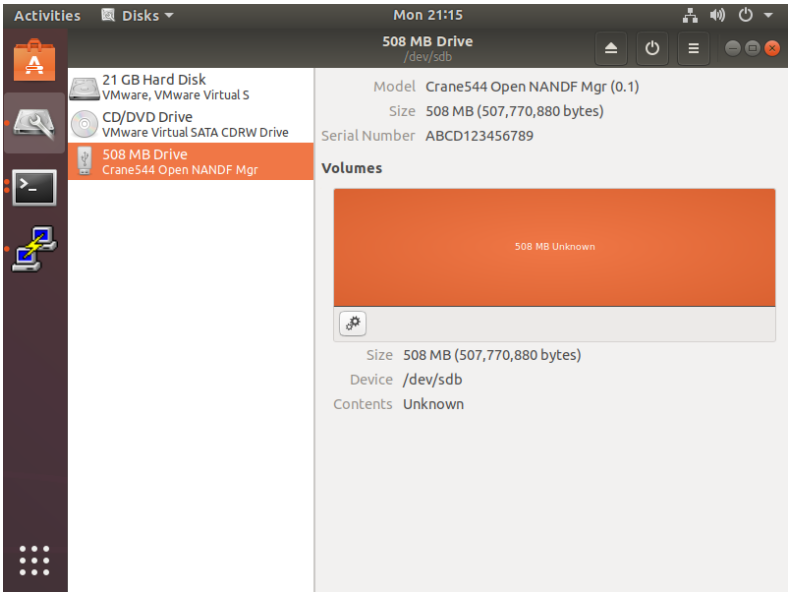
7. Open a new terminal and navigate to the location where the above files are placed. In this case, I am navigating to desktop using cd Desktop command in the new terminal. Always run the terminal as root user.
8. Now type the command "**./download.sh**" from the desktop location as shown below. Update the bin folder name in the download.sh file if there are any changes to the file name.



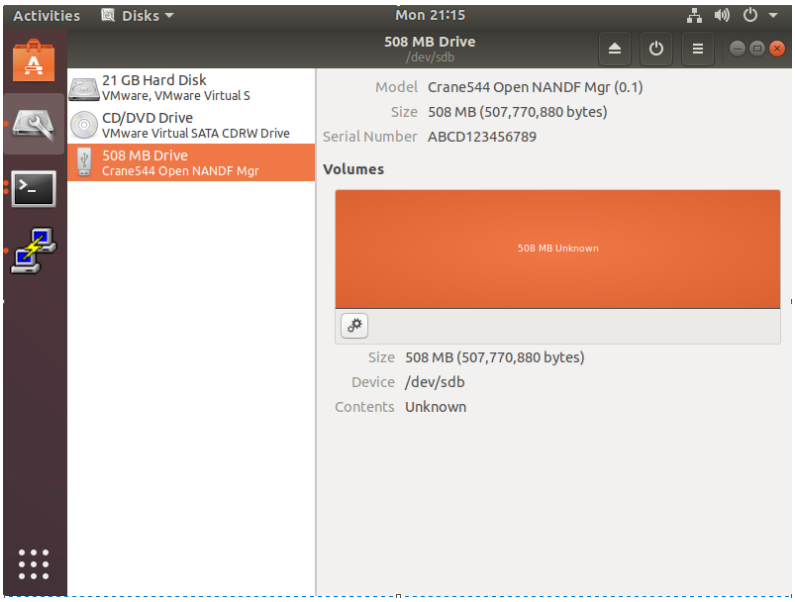
9. If the command is successfully executed, successfully mounted message is displayed on the putty console terminal as shown below



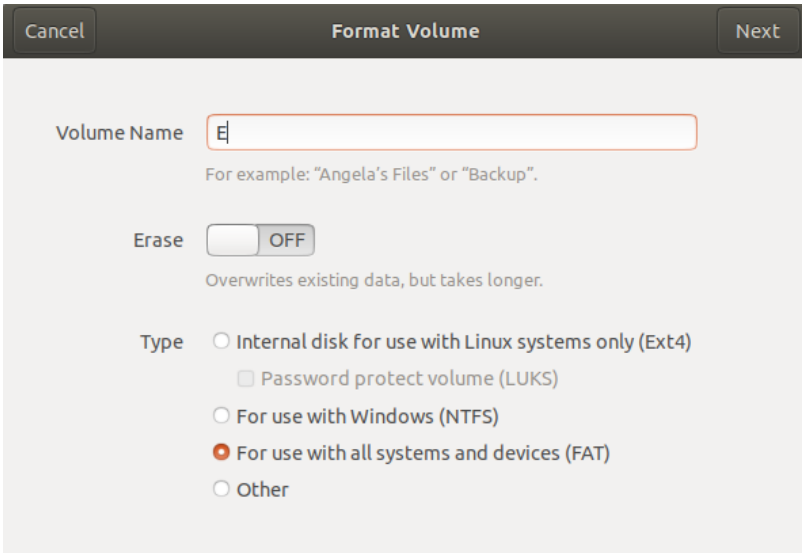
10. Removable Drive is detected under the disks folder as shown in the below image



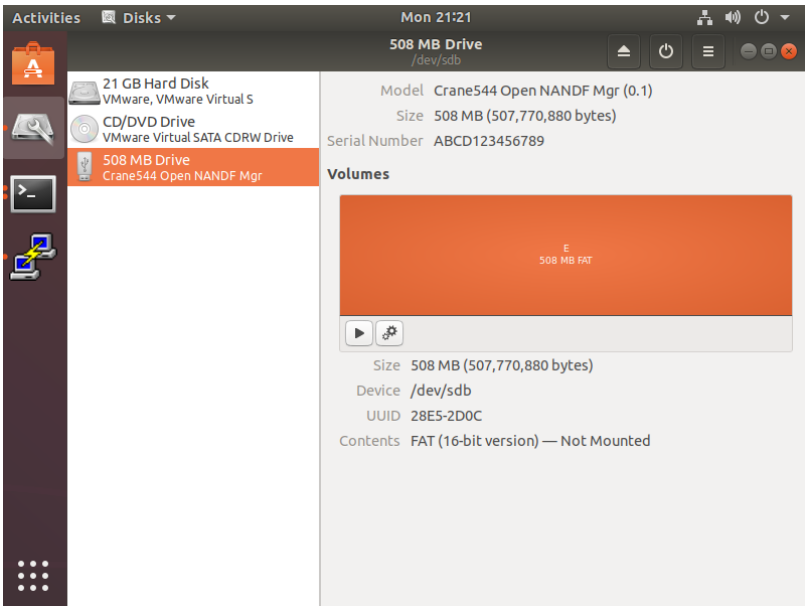
11. Now the disk needs to be formatted before it is used by clicking on the settings button



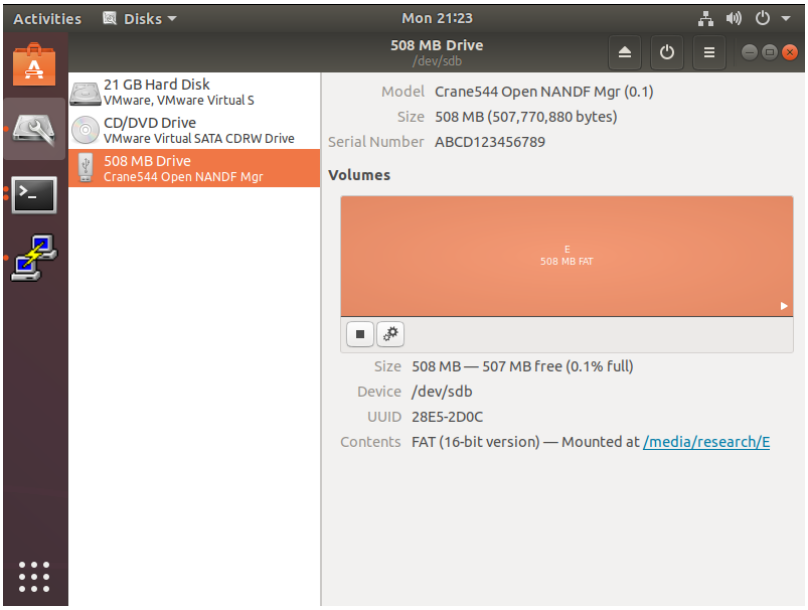
12. You will be prompted to enter Volume name of the disk as shown below and click on next and then click Format



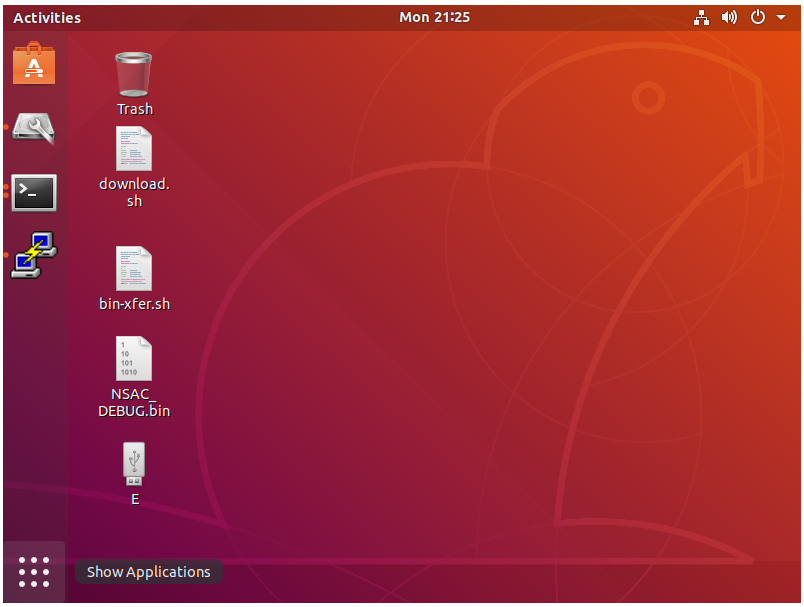
13. Now the disk needs to be started by clicking on the play button available on the disk as shown below



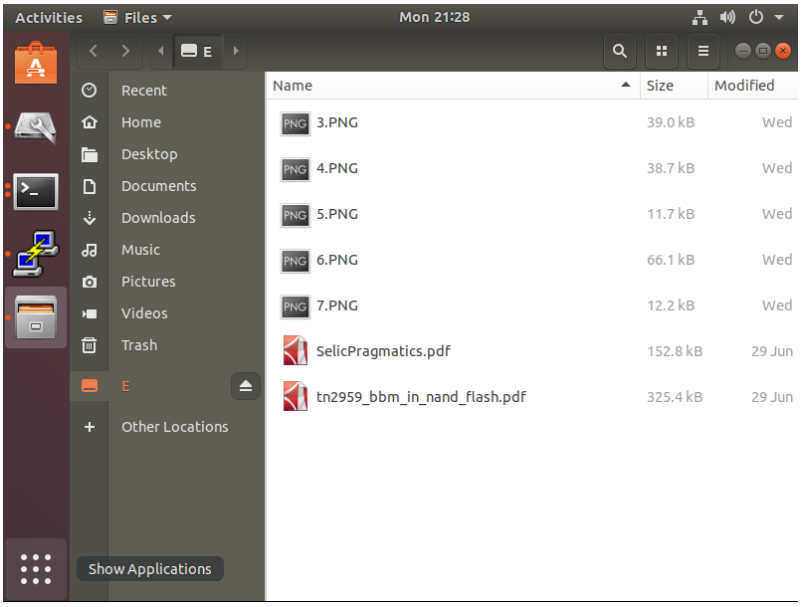
14. Once the disk is started successfully, the status of the drive looks as below



15. Disk with the volume name given while formatting on the desktop as shown below



16. Open the E disk and the disk is ready to write files and used as any other USB Flash device for reading and writing as shown below

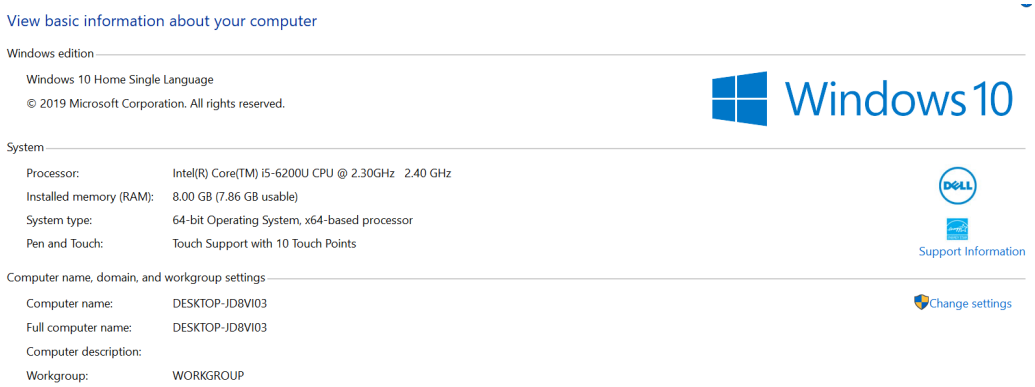


17. To remove the drive, click on the disconnect usb and the PuTTY connection is automatically closed.

Note: The OpenNFM testbed setup is done for Windows 10 operating system for compiling code and generating Binary file and the generated Binary file is used in Ubuntu VM installed in the same windows machine

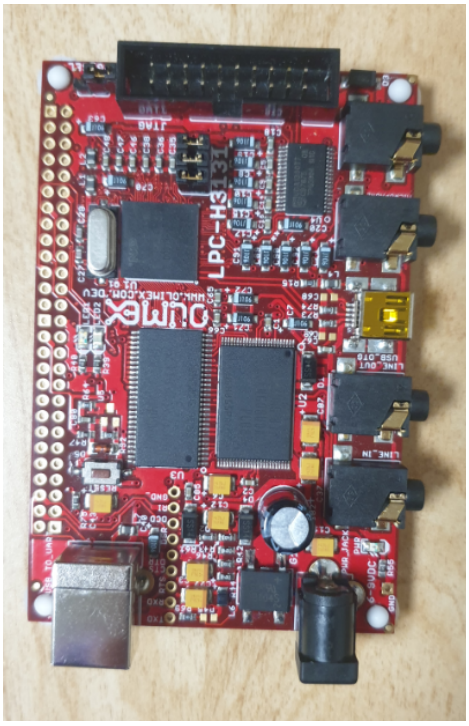
IV. Equipment:

a. Desktop:



b. USB A to B & A to Mini cables:

c. LPC-H3131 Prototype Board:



References

[1] Niusen Chen, Bo Chen, and Weisong Shi. A Cross-layer Plausibly Deniable Encryption System for Mobile Devices. 18th EAI International Conference on Security and Privacy in Communication Networks (SecureComm '22), 2022.

[2] Wen Xie, Niusen Chen, and Bo Chen. Enabling Accurate Data Recovery for Mobile Devices against Malware Attacks. 18th EAI International Conference on Security and Privacy in Communication Networks (SecureComm '22), 2022.

[3] Niusen Chen, and Bo Chen. Duplicates also Matter! Towards Secure Deletion on Flash-based Storage Media by Removing Duplicates. The 17th ACM ASIA Conference on Computer and Communications Security (ASIACCS '22), 2022.

[4] Niusen Chen, and Bo Chen. Defending against OS-level Malware in Mobile Devices via Real-time Malware Detection and Storage Restoration. Journal of Cybersecurity and Privacy 2, no. 2 (2022): 311-328.

[5] Niusen Chen, Bo Chen, and Weisong Shi. The Block-based Mobile PDE Systems Are Not Secure – Experimental Attacks. 2022 EAI International Conference on Applied Cryptography in Computer and Communications (AC3 '22), 2022.

[6] Niusen Chen, Wen Xie, and Bo Chen. Combating the OS-level Malware in Mobile Devices by Leveraging Isolation and Steganography. The Second ACNS Workshop on Secure Cryptographic Implementation (SCI '21), 2021.

[7] Bo Chen, and Niusen Chen. A Secure Plausibly Deniable System for Mobile Devices against Multi-snapshot Adversaries. 2020 IEEE Symposium on Security and Privacy (S&P '20), 2020.

[8] Wen Xie, Niusen Chen, and Bo Chen. Incorporating Malware Detection into The Flash Translation Layer. 2020 IEEE Symposium on Security and Privacy (S&P '20), 2020.

[9] Peiying Wang, Shijie Jia, Bo Chen, Luning Xia and Peng Liu. MimosaFTL: Adding Secure and Practical Ransomware Defense Strategy to Flash Translation Layer. The Ninth ACM Conference on Data and Application Security and Privacy (CODASPY '19), 2019.

[10] Le Guan, Shijie Jia, Bo Chen, Fengwei Zhang, Bo Luo, Jingqiang Lin, Peng Liu, Xinyu Xing, and Luning Xia. Supporting Transparent Snapshot for Bare-metal Malware Analysis on Mobile Devices. 2017 Annual Computer Security Applications Conference (ACSAC '17), 2017.

[11] Shijie Jia, Luning Xia, Bo Chen, and Peng Liu. DEFTL: Implementing Plausibly Deniable Encryption in Flash Translation Layer. 2017 ACM Conference on Computer and Communications Security (CCS '17), 2017.

[12] Bo Chen, Shijie Jia, Luning Xia, and Peng Liu. Sanitizing Data is Not Enough! Towards Sanitizing Structural Artifacts in Flash Media. 2016 Annual Computer Security Applications Conference (ACSAC '16), 2016.

[13] Shijie Jia, Luning Xia, Bo Chen, and Peng Liu. NFPS: Adding Undetectable Secure Deletion to Flash Translation Layer. The 11th ACM Asia Conference on Computer and Communications Security (ASIACCS '16), 2016.

[14] Bo Chen, and Radu Sion. HiFlash: A history independent flash device. arXiv preprint arXiv:1511.05180, 2015.

[15] Lpc-h3131. Retrieved June 29, 2022, from https://www.olimex.com/Products/ARM/NXP/LPC-H3131/.

[16] Opennfm. Retrieved June 29, 2022, from https://code.google.com/p/opennfm/.

[17] Tankasala, Deepthi, Niusen Chen, and Bo Chen. Step-by-step Guideline for Creating A Testbed for Flash Memory Research via LPC-H3131 and OpenNFM. Technical report, MTU CS Department, 2020.

Appendix

bin-xfer.sh

```
#!/bin/sh
INFILE=/dev/null
OUTFILE=/dev/null

exists() {
  command -v $1 >/dev/null 2>&1
```

```
}

while [ $# -gt 0 ]; do
 case "$1" in
 -i)
 shift
 INFILE="$1"
 ;;
 -o)
 shift
 OUTFILE="$1"
 ;;
 -h|--help)
 echo "$0 -i infile -o outfile"
 ;;
 *)
 INFILE="$1"
 esac
 shift
done
cat << EOF
binary-xfer utility for minicom
Sending file ${INFILE} to ${OUTFILE}
EOF

if (exists pv); then
  pv --force -i 0.25 -B 128  ${INFILE}  2>&1 > ${OUTFILE}
else
  cat ${INFILE} > ${OUTFILE}
fi

cat << EOF

File transfer complete
EOF

sleep 1
```

download.sh

```
sudo ./bin-xfer.sh -i onfm_fresh.bin -o /dev/ttyUSB0
```