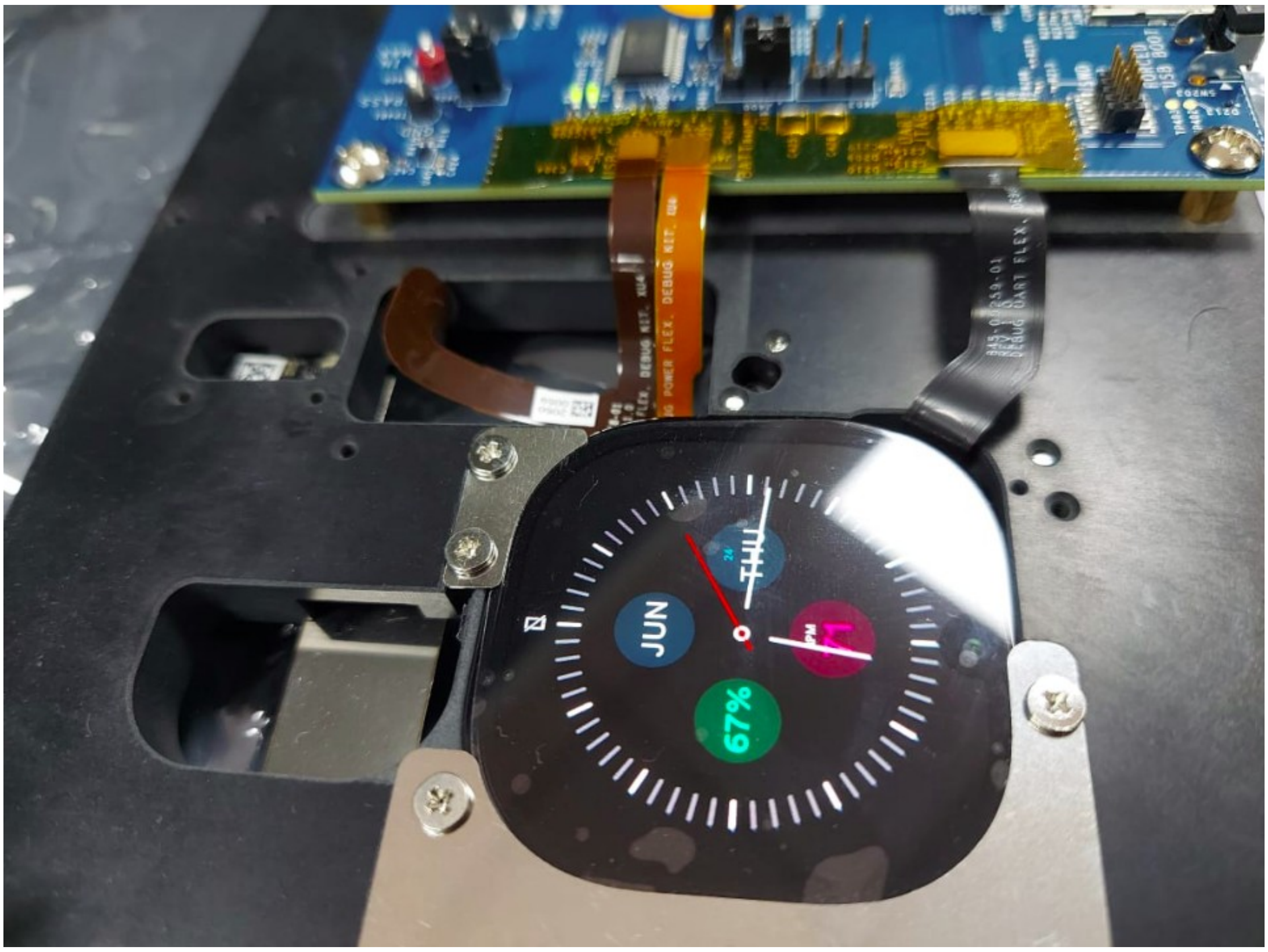
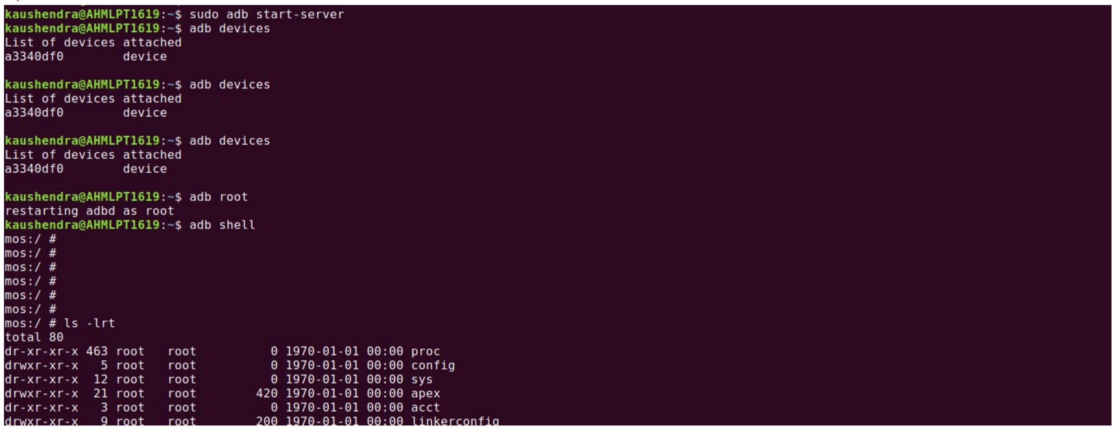
## CROSS COMPILE TEST BINARY FOR ARMV7 NED BOARD

Step 1: Connect USB-Type-B with Power Socket and USB-Type-C to Host Machine and press power button to turn on the NED board.



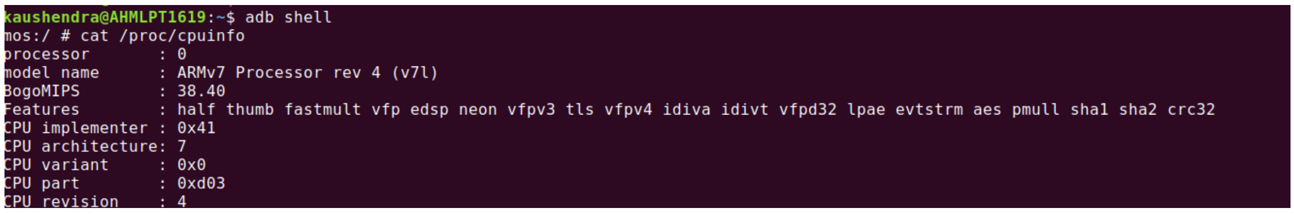
Step 2:get the adb of the Board using below commands.

$sudo adb start-server
$adb devices
$adb root
$adb shell



```
kaushendra@AHMLPT1619:~$ sudo adb start-server
kaushendra@AHMLPT1619:~$ adb devices
List of devices attached
a3340df0        device

kaushendra@AHMLPT1619:~$ adb devices
List of devices attached
a3340df0        device

kaushendra@AHMLPT1619:~$ adb devices
List of devices attached
a3340df0        device

kaushendra@AHMLPT1619:~$ adb root
restarting adbd as root
kaushendra@AHMLPT1619:~$ adb shell
mos:/ #
mos:/ #
mos:/ #
mos:/ #
mos:/ #
mos:/ #
mos:/ # ls -lrt
total 80
dr-xr-xr-x 463 root   root          0 1970-01-01 00:00 proc
drwxr-xr-x   5 root   root          0 1970-01-01 00:00 config
dr-xr-xr-x  12 root   root          0 1970-01-01 00:00 sys
drwxr-xr-x  21 root   root        420 1970-01-01 00:00 apex
dr-xr-xr-x   3 root   root          0 1970-01-01 00:00 acct
drwxr-xr-x   9 root   root        200 1970-01-01 00:00 linkerconfig
```

Step 3: understand the architecture of the board using cpuinfo command from adb shell.
$ adb shell
$ cat /proc/cpuinfo



```
kaushendra@AHMLPT1619:~$ adb shell
mos:/ # cat /proc/cpuinfo
processor       : 0
model name      : ARMv7 Processor rev 4 (v7l)
BogoMIPS        : 38.40
Features        : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt vfpd32 lpae evtstrm aes pmull sha1 sha2 crc32
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4
```

Step 4: Generate the Armv7 toolchain for board specific:

[I] First thing you need Android NDK :if you dont have then grab from the below steps run on your machine using NDK.
**$curl -O** \http://dl.google.com/android/repository/android-ndk-r12b-linux-x86_64.zip
**$unzip android-ndk-r13b-linux-x86_64.zip**

[II] Build your custom toolchain:
$cd android-ndk-r12b/build/tools
$./make_standalone_toolchain.py --arch arm --api 24 --install-dir=my-toolchain

[III] Develop python based script which compiles a C library/binary for board specific architecture.

- implementing  build.sh

```
#armv7 set-path
CC=/home/kaushendra/kaush/ARROW/FRL/andriod_studio/android-ndk-r12b/build/tools/my-toolchain/bin/arm-linux-androideabi-gcc
AR=/home/kaushendra/kaush/ARROW/FRL/andriod_studio/android-ndk-r12b/build/tools/my-toolchain/bin/arm-linux-androideabi-ar
SYSROOT=/home/kaushendra/kaush/ARROW/FRL/andriod_studio/android-ndk-r12b/platforms/android-24/arch-arm
```

```
INCS=-I/home/kaushendra/kaush/ARROW/FRL/andriod_studio/android-ndk-r12b/platforms/android-
24/arch-arm/usr/include/android

#remove old files
rm -rf sensor.o sensor_test.out libsensor.a build/

#build libLegacy
$CC -fPIE -c sensor.c -o sensor.o --sysroot=$SYSROOT $INCS -llog
$AR rcs libsensor.a sensor.o

#build dynamic library: libBridge
$CC -fPIE -c sensor_test.c -o sensor_test.o --sysroot=$SYSROOT -I./native $INCS
$CC -pie sensor_test.o ./libsensor.a -o sensor_test.out --sysroot=$SYSROOT -llog

#copy library to build folder
if [ ! -d build ]; then
    mkdir build
fi
cp sensor_test.out ./build
```

- implementing test_validate_sensor_data.py

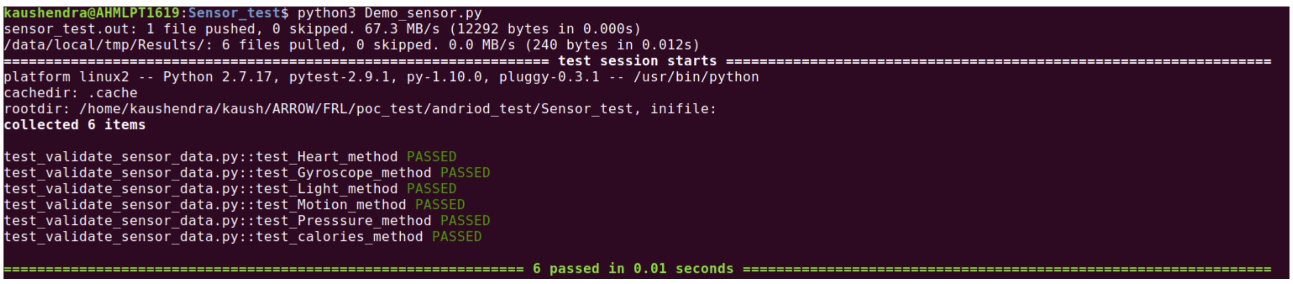
```
import subprocess
from ctypes import *
import pytest
import sys
import os
import linecache
PATH = os.getcwd()

Heart_sensor_filepath = "./Results/testHeartSensorSensor.txt"
Gyroscope_sensor_filepath = "./Results/testGyroscopeSensor.txt"
Light_sensor_filepath = "./Results/testLightSensor.txt"
Motion_sensor_filepath = "./Results/testMotionSensor.txt"
Presssure_sensor_filepath = "./Results/testPressureSensor.txt"
calories_sensor_filepath = "./Results/testCaloriesSensor.txt"


def test_Heart_method():
        file = open(Heart_sensor_filepath,'r+')
        particular_line = linecache.getline(Heart_sensor_filepath, 2)
        assert particular_line == "89\n","test failed"
        file.close


def test_Gyroscope_method():
        file = open(Gyroscope_sensor_filepath,'r+')
        particular_line = linecache.getline(Gyroscope_sensor_filepath, 2)
        assert particular_line == "-10 50 0\n","test failed"
        file.close


def test_Light_method():
        file = open(Light_sensor_filepath,'r+')
        particular_line = linecache.getline(Light_sensor_filepath, 2)
        assert particular_line == "50.000000\n","test failed"
        file.close


def test_Motion_method():
        file = open(Motion_sensor_filepath,'r+')
        particular_line = linecache.getline(Motion_sensor_filepath, 2)
        assert particular_line == "250 500 1000\n","test failed"
        file.close


def test_Presssure_method():
        file = open(Presssure_sensor_filepath,'r+')
        particular_line = linecache.getline(Presssure_sensor_filepath, 2)
        assert particular_line == "30.600000\n","test failed"
        file.close


def test_calories_method():
        file = open(calories_sensor_filepath,'r+')
        particular_line = linecache.getline(calories_sensor_filepath, 2)
        assert particular_line == "0.090000\n","test failed"
        file.close
```

- implementing  Demo_sensor.py

```
import subprocess

#simple python running
subprocess.call("./build.sh",shell=True)
subprocess.call("adb shell rm -f /data/local/tmp/sensor_test.out",shell=True)
subprocess.call("adb push sensor_test.out /data/local/tmp/.",shell=True)

subprocess.call("adb shell ./data/local/tmp/sensor_test.out Gyroscope",shell=True)
subprocess.call("adb shell ./data/local/tmp/sensor_test.out HeartRate",shell=True)
subprocess.call("adb shell ./data/local/tmp/sensor_test.out VirtualCalorie",shell=True)
subprocess.call("adb shell ./data/local/tmp/sensor_test.out Presssure",shell=True)
subprocess.call("adb shell ./data/local/tmp/sensor_test.out Light",shell=True)
subprocess.call("adb shell ./data/local/tmp/sensor_test.out Motion",shell=True)
subprocess.call("rm -rf ./Results", shell=True)
subprocess.call("adb pull /data/local/tmp/Results ./", shell=True)

#subprocess.call("adb shell rm -f /data/local/tmp/sensor_test.out",shell=True)
#subprocess.call("adb shell rm -r /data/local/tmp/Results", shell=True)

subprocess.call("py.test test_validate_sensor_data.py -v", shell=True)
```

Step 5: Runnig the Cross-Compiled Library and binary over NED Board and checking Test Results:

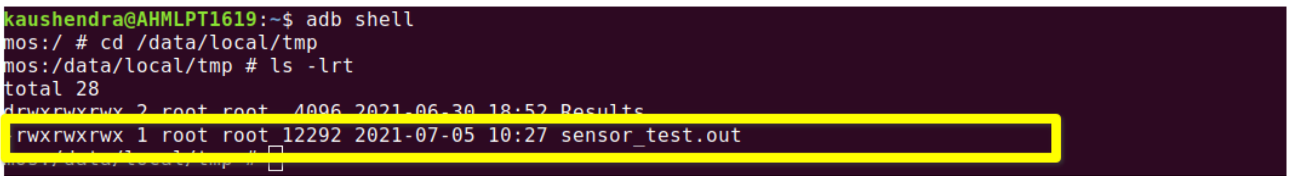
$python3 Demo_sensor.py



```
kaushendra@AHMLPT1619:Sensor_test$ python3 Demo_sensor.py
sensor_test.out: 1 file pushed, 0 skipped. 67.3 MB/s (12292 bytes in 0.000s)
/data/local/tmp/Results/: 6 files pulled, 0 skipped. 0.0 MB/s (240 bytes in 0.012s)
================================================================= test session starts ==================================================================
platform linux2 -- Python 2.7.17, pytest-2.9.1, py-1.10.0, pluggy-0.3.1 -- /usr/bin/python
cachedir: .cache
rootdir: /home/kaushendra/kaush/ARROW/FRL/poc_test/andriod_test/Sensor_test, inifile:
collected 6 items

test_validate_sensor_data.py::test_Heart_method PASSED
test_validate_sensor_data.py::test_Gyroscope_method PASSED
test_validate_sensor_data.py::test_Light_method PASSED
test_validate_sensor_data.py::test_Motion_method PASSED
test_validate_sensor_data.py::test_Presssure_method PASSED
test_validate_sensor_data.py::test_calories_method PASSED

============================================================== 6 passed in 0.01 seconds ===============================================================
```

Step 6: Checking binary over NED Board :

$adb shell
$cd /data/local/tmp/



```
kaushendra@AHMLPT1619:~$ adb shell
mos:/ # cd /data/local/tmp
mos:/data/local/tmp # ls -lrt
total 28
drwxrwxrwx 2 root root  4096 2021-06-30 18:52 Results
-rwxrwxrwx 1 root root 12292 2021-07-05 10:27 sensor_test.out
mos:/data/local/tmp #
```