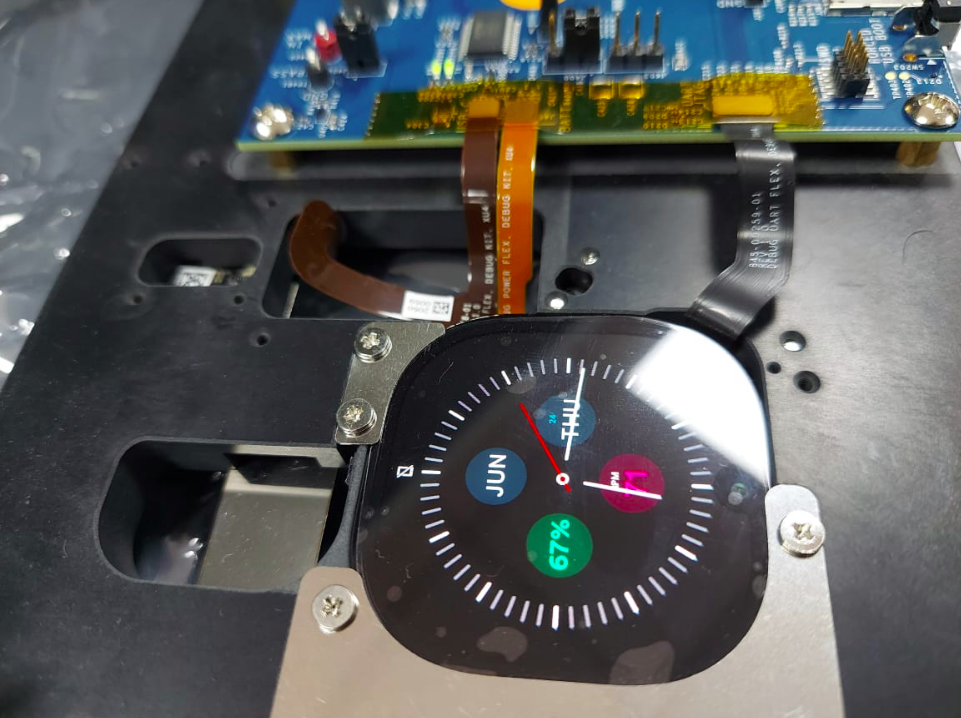
**Cross Compile TEST Binary for ARMv7 NED BoArd**

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.



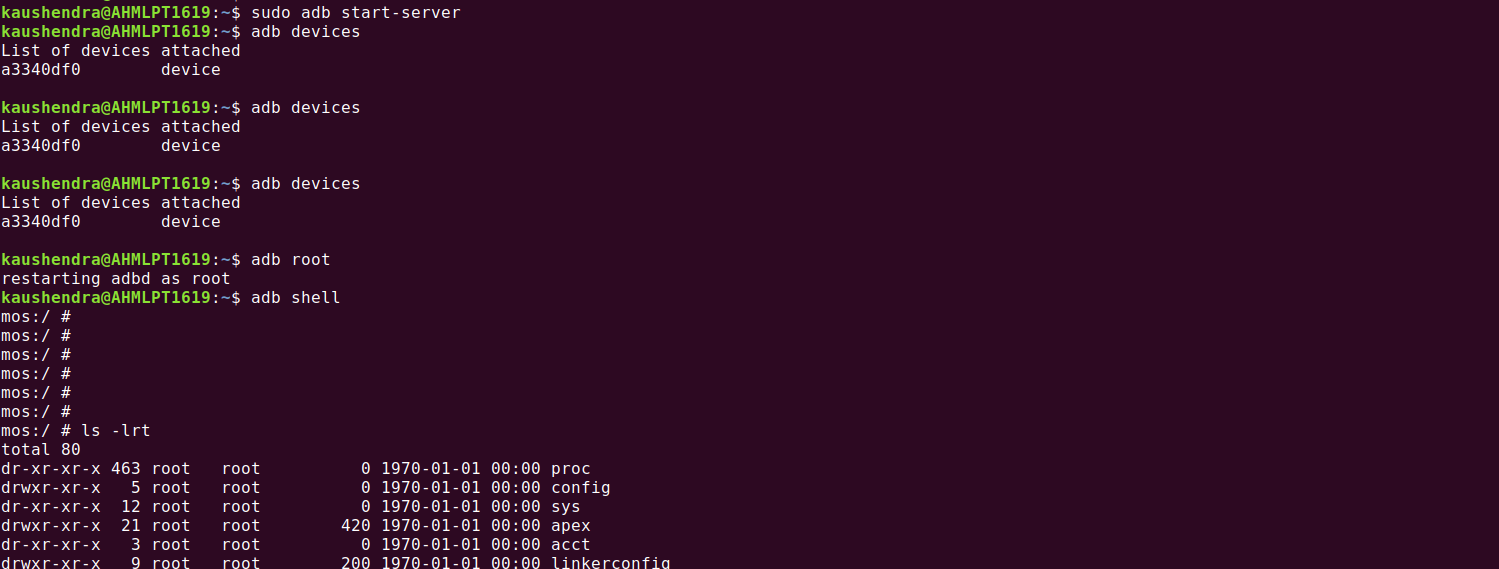
Step 2:get the adb of the Board using below commands.

$sudo adb start-server

$adb devices

$adb root

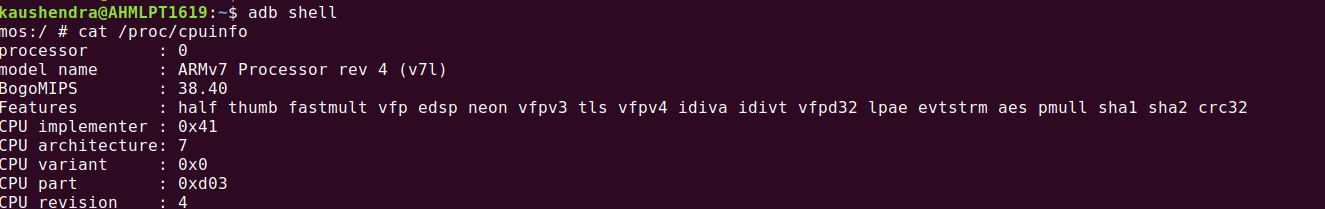
$adb shell



Step 3: understand the architecture of the board using cpuinfo command from adb shell.

$ adb shell

$ cat /proc/cpuinfo



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](https://developer.android.com/studio/projects/install-ndk" \l "install_ndk_and_cmake_automatically).

**$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

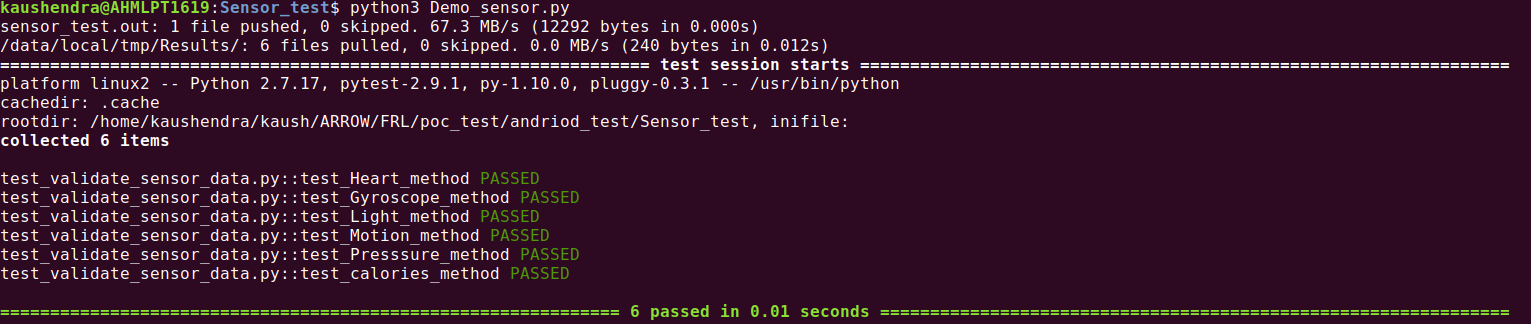
|  |
| --- |
| import subprocess  from ctypes import \*  import pytest  import sys  import os  import linecache  PATH = os.getcwd()  Heart\_sensor\_filepath = "./Results/testHeartSensorSensor.txt"  Gyroscope\_sensor\_filepath = "./Results/testGyroscopeSensor.txt"  Light\_sensor\_filepath = "./Results/testLightSensor.txt"  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



Step 6: Checking binary over NED Board :

$adb shell

$cd /data/local/tmp/

