Design & Development of Android Based IoT Applications using Qualcomm Snapdragon board
A Practical Approach

Experimental Manual for B.Tech & M.Tech Students

Designed By: Ms. Jasleen Kaur, PhD Scholar (CSE)

Under the Guidance of: Dr. SRN Reddy, Associate Professor and Head (CSE)

Computer Science & Engineering Department

Indira Gandhi Delhi Technical University for Women

Kashmere Gate, Delhi-110006

INDEX

INTRODUCTION

Snapdragon is a family of mobile system on a chip (SoC) by Qualcomm. Qualcomm considers Snapdragon a "platform" for use in smart phones, tablets, and smartbook devices. The original Snapdragon CPU namely **SCORPION** is Qualcomm's own design similar to those of the ARM Cortex-A8 core and it is based on the ARMv7 instruction set. It has much higher performance for multimedia-related SIMD operations. The successor to Scorpion, found in S4 Snapdragon SoCs, is named **Krait** CPU. It has many similarities with the ARM Cortex-A15 CPU and is based on the ARMv7 instruction set. The majority of Snapdragon processors contain the circuitry to decode high-definition video (HD) resolution at 720p or 1080p depending on the Snapdragon chip. Adreno, the company's proprietary GPU series, integrated into Snapdragon chips (and certain other Qualcomm chips) is Qualcomm's own design, using assets the company acquired from AMD. The Adreno 225 GPU in Snapdragon S4 SoCs adds support for DirectX 9/Shader Model 3.0, which makes it compatible with Microsoft's Windows 8. All Snapdragons feature one or more DSPs called Hexagon namely QDSP5 or QDSP6 in the modem and multimedia parts. In the Snapdragon 200–800 series, one of the multimedia Hexagons is programmable by the users through the Hexagon SDK. The multimedia Hexagons are mostly used for audio encoding/decoding, the newer Snapdragons have a hardware block called Venus for video encoding/decoding. Snapdragon SoC also have on-die Wi-Fi, GPS/GLONASS and Bluetooth basebands.[1]

For this Lab Project we have used the Snapdragon Dragon Board 410c.

The Dragon Board 410c (‘410c’) board is a board based on Qualcomm Snapdragon 400 series of SoC’s.

The following table lists its key features:

Processor	Qualcomm Snapdragon 410 Quad-core ARM® Cortex® A53 at up to 1.2 GHz per core 64-Bit capable Qualcomm Adreno 306 400MHz GPU for PC-class graphics with support for advanced APIs, including OpenGL ES 3.0, OpenCL, DirectX, and content security
Memory/ Storage	1GB LPDDR3 533MHz 8GB eMMC 4.51 SD 3.0 (UHS-I)
Video	1080p@30fps HD video playback and capture with H.264 (AVC), and 720p playback with H.265 (HEVC)
Camera Support	Integrated ISP with support for image sensors up to 13MP
Audio	PCM/AAC+/MP3/WMA, ECNS, Audio+ post-processing (optional)
Connectivity	WLAN 802.11 b/g/n 2.4GHz Bluetooth 4.1 One USB 2.0 micro B (device mode only) Two USB 2.0 (host mode only)
User Interface	Power/Reset Volume Up/down 6 LED indicators • 4 - user controllable • 2 - for radios (BT and WLAN activity)
OS-support	Android 5.1 Linux based on Debian Windows 10 IoT core
Power, Mechanical and Environmental	Power: +6.5V to +18V Dimensions: 54mm by 85mm meeting 96Boards™ Consumer Edition standard dimensions specifications. Operating Temp: 0°C to +70°C RoHS and Reach compliant

I/O Interfaces	One 40-pin Low Speed (LS) expansion connector • UART, SPI, I2S, I2C x2, GPIO x12, DC power One 60-pin High Speed (HS) expansion connector • 4L-MIPI DSI, USB, I2C x2, 2L+4L-MIPI CSI Footprint for one optional 16-pin analog expansion connector for stereo headset/ line-out, speaker and analog line-in The board can be made compatible with Arduino using an add-on mezzanine board
External Storage	Micro SD card slot

Table 1: Features of Snapdragon Dragon Board 410c [2]

Before you power up your 410c board for the first time you will need the following:

- 410c board.
- A 12V power supply Adapter.
- A HDMI or DVI LCD Monitor that supports a resolution of 1080P/30Hz.
- HDMI-HDMI cable or HDMI-DVI cable to connect the board to the Monitor.
- A computer keyboard with USB interface
- A computer mouse with USB interface.

Board Overview:

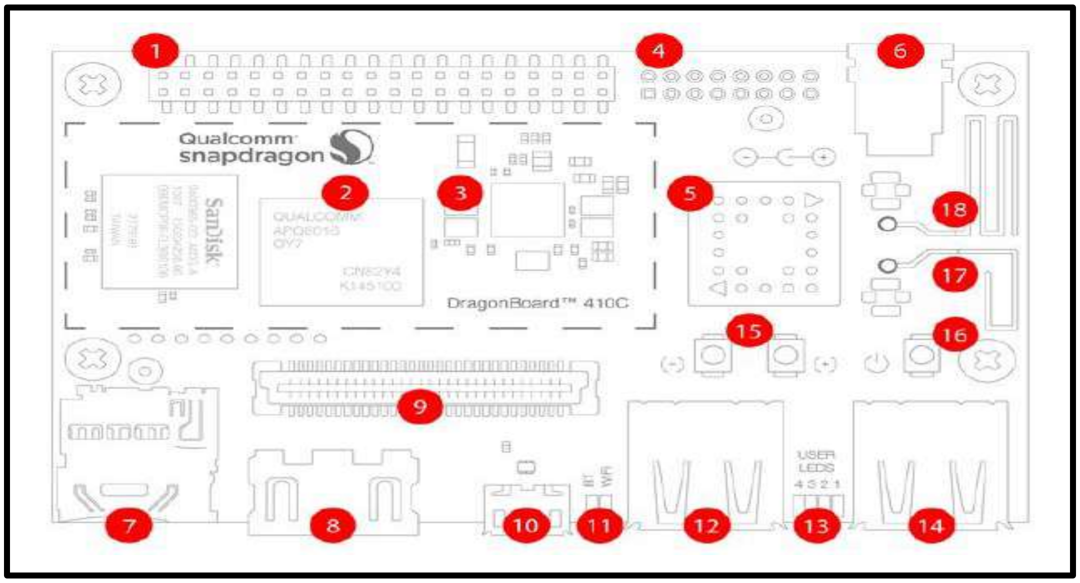


Figure 1: Dragon Board 410c Front side [2]

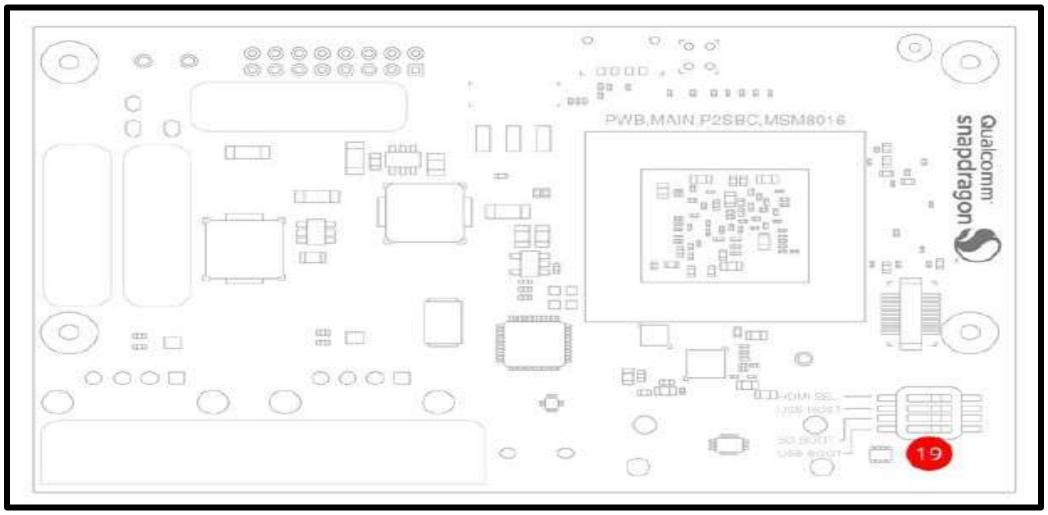
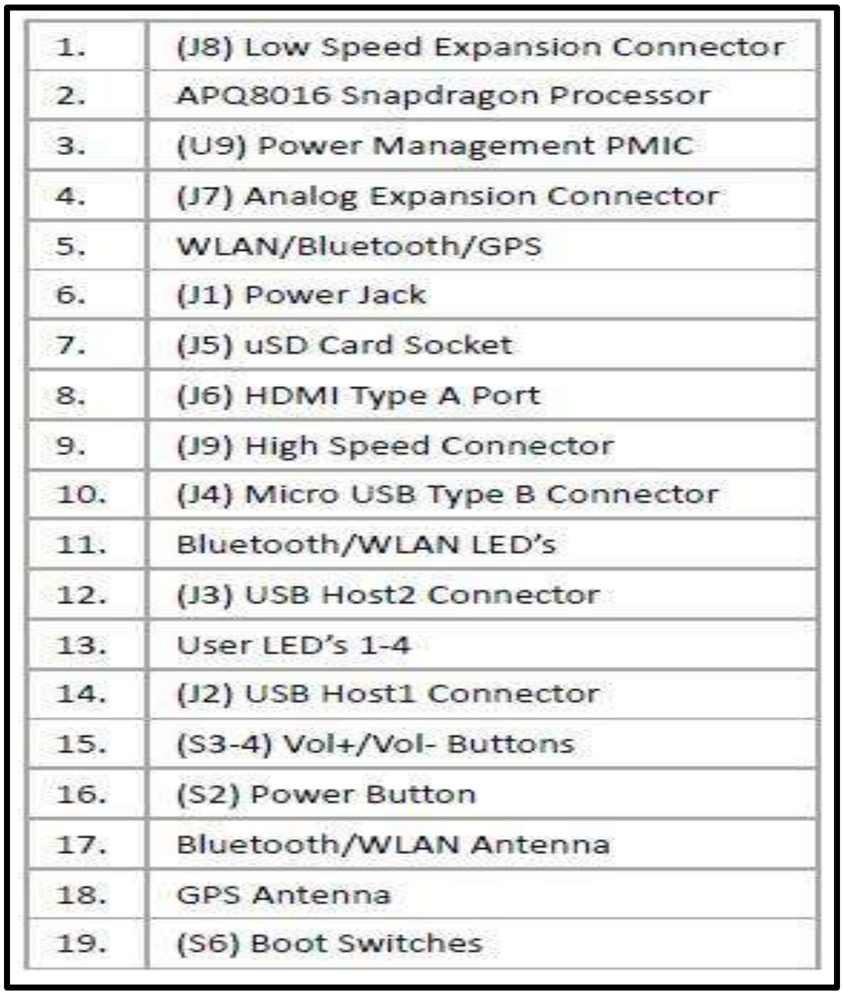


Figure 2: Dragon Board 410c Backside [2]



1.	(J8) Low Speed Expansion Connector
2.	APQ8016 Snapdragon Processor
3.	(U9) Power Management PMIC
4.	(J7) Analog Expansion Connector
5.	WLAN/Bluetooth/GPS
6.	(J1) Power Jack
7.	(J5) uSD Card Socket
8.	(J6) HDMI Type A Port
9.	(J9) High Speed Connector
10.	(J4) Micro USB Type B Connector
11.	Bluetooth/WLAN LED's
12.	(J3) USB Host2 Connector
13.	User LED's 1-4
14.	(J2) USB Host1 Connector
15.	(S3-4) Vol+/Vol- Buttons
16.	(S2) Power Button
17.	Bluetooth/WLAN Antenna
18.	GPS Antenna
19.	(S6) Boot Switches

Table 2: Refer with Fig 1 and Fig 2. [2]

EXPERIMENT-1

AIM: Introduction to Snapdragon Dragon Board 410c

REQUIREMENTS:

1. Snapdragon Board 410c
2. TFT
3. HDMI cable
4. Power Supply

THEORY:

Dragon board 410c:

- The Dragon Board 410c is the first development board based on a Qualcomm Snapdragon 400 series processor.
- Based on the 64-bit capable Snapdragon 410 processor, the DragonBoard 410c is designed to support rapid software development, education and prototyping.
- It is powerful, feature- rich, versatile and easy to use exposed board platform for component vendors, software and embedded developers.
- The DragonBoard has 1GB RAM and 8GB eMMC Flash built in memory.

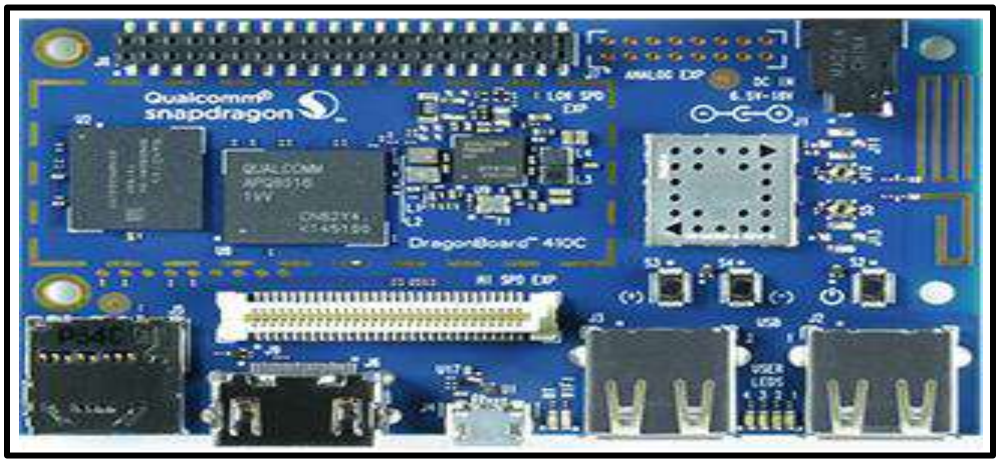


Figure 1.1: Snapdragon board410c

Features of dragonboard410c

- OS Support: Android 5.1 (Lollipop) on Linux Kernel 3.10, Linux based on Debian 8.0, and Windows 10 IoT Core.
- CPU: Quad-core ARM® Cortex® A53 at up to 1.2 GHz per core with both 32-bit and 64-bit support.
- **I/O Interfaces:** HDMI Full-size Type A connector, one micro USB (device mode only), two USB 2.0 (host mode only), micro SD card slot.
- Connectivity and Location:
    - Wi-Fi 802.11 b/g/n 2.4GHz, integrated digital core
    - Bluetooth 4.1, integrated digital core
    - Qualcomm® IZat™ location technology Gen8C
    - On-board Wi-Fi, BT and GPS antenna.

TFT touch screen:

- A TFT touch screen is a combination device that includes a TFT LCD display and a touch technology overlay on the screen. The device can both display content and act as an interface device for whoever is using it.
- To display the user interface, Touchscreen is being used so as to get input and display output.

Fig 1.2: TFT touch screen

INTERFACING DIAGRAM:

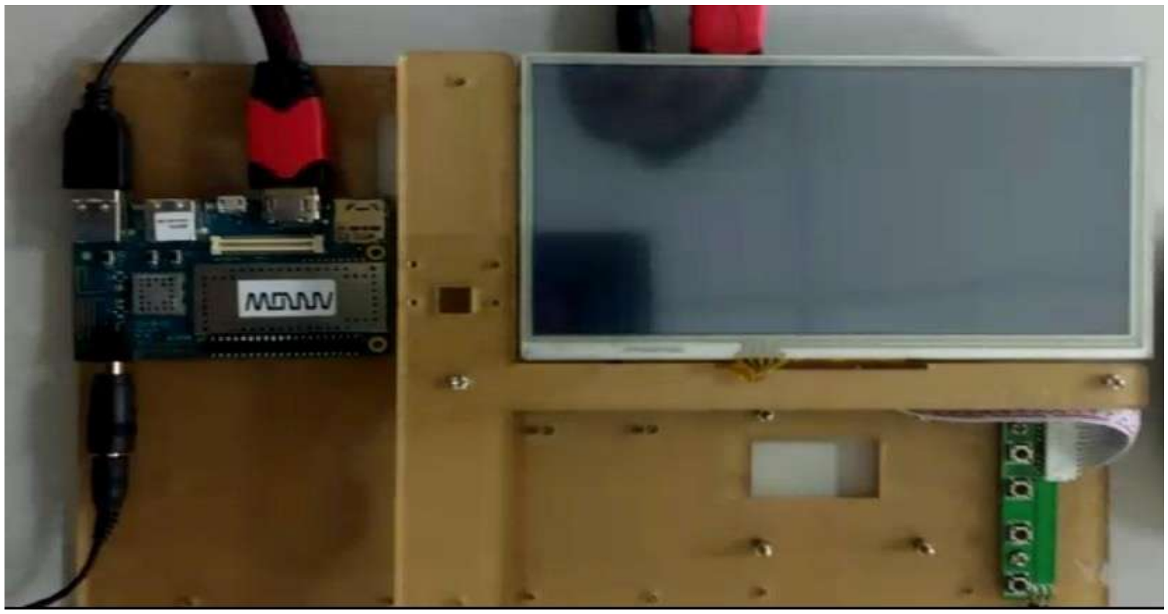


Figure 1.3: Interfacing of Snapdragon with TFT

PROCEDURE:

- First of all connect the TFT screen to dragon board with the help of HDMI cable.
- Connect the charger to both of the Dragon Board and the TFT screen and then give the power supply.
- Now turn on the power button of Dragon Board. After the power button on the TFT screen will be on.

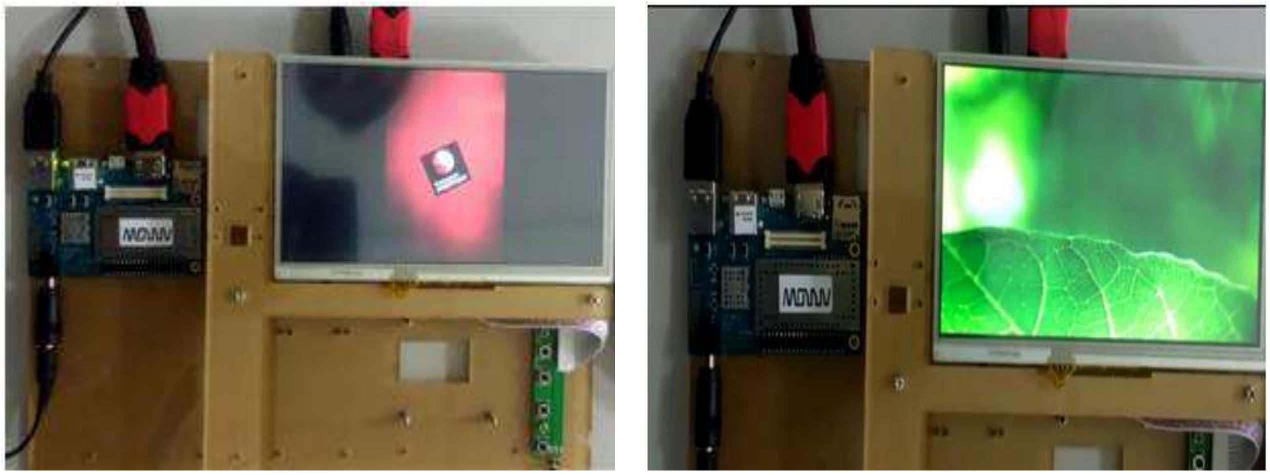


Figure 1.4: Turning on the Dragon Board and TFT screen

- Now swipe up the lock of screen.

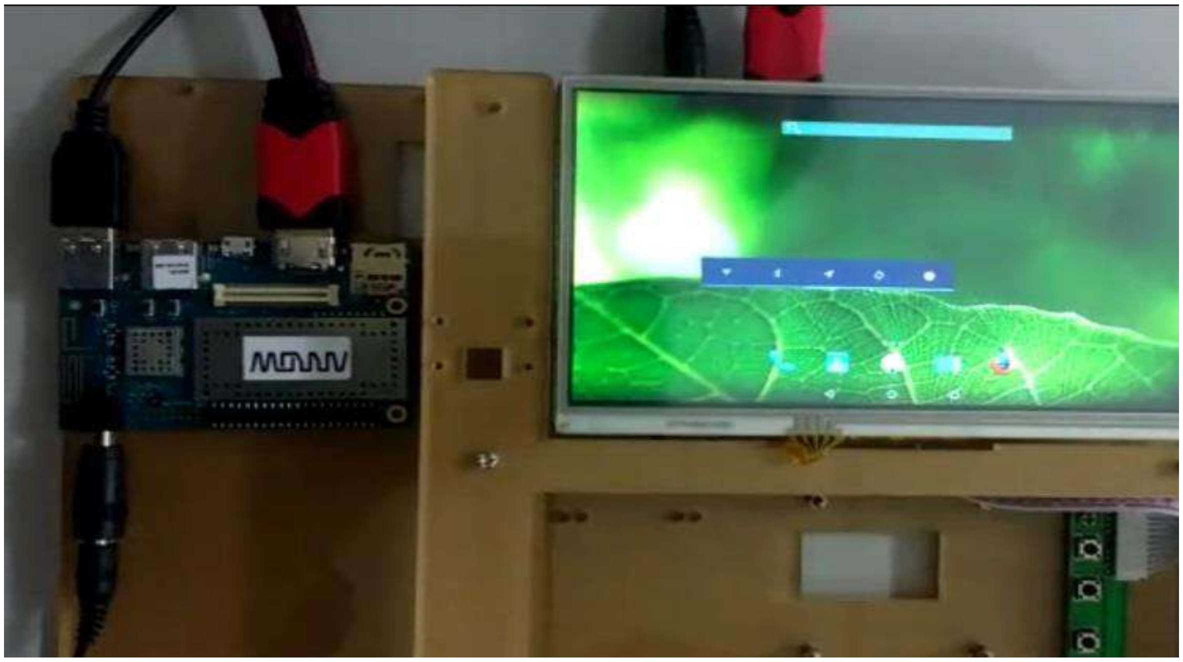


Figure 1.5: Displaying home screen of Dragon board on TFT screen

CONCLUSION:

The snapdragon board has been successfully interfaced with TFT screen.

EXPERIMENT-2

AIM: To install Android Studio and design a basic Welcome application. Deploy the application on the Snapdragon DragonBoard 410c.

COMPONENTS REQUIRED:

Hardware Components: Snapdragon DragonBoard 410c, HDMI-to HDMI cable, 2 AC adapters (12 V), TFT screen, Driver Board for TFT screen, mouse, Micro USB cable, laptop/PC.

Software Components: Android Studio or any other such IDE installed on the PC/Laptop.

THEORY:

Android Studio is the official integrated development environment (IDE) for Android platform development. It was announced on May 16, 2013 at the Google I/O conference. Android Studio is freely available under the Apache License 2.0.

Based on JetBrains' IntelliJ IDEA software, Android Studio is designed specifically for Android development. It is available for download on Windows, Mac OS X and Linux, and replaced Eclipse Android Development Tools (ADT) as Google's primary IDE for native Android application development.

Features of Android Studio:

New features are expected to be rolled out with each release of Android Studio. The following features are provided in the current stable version:

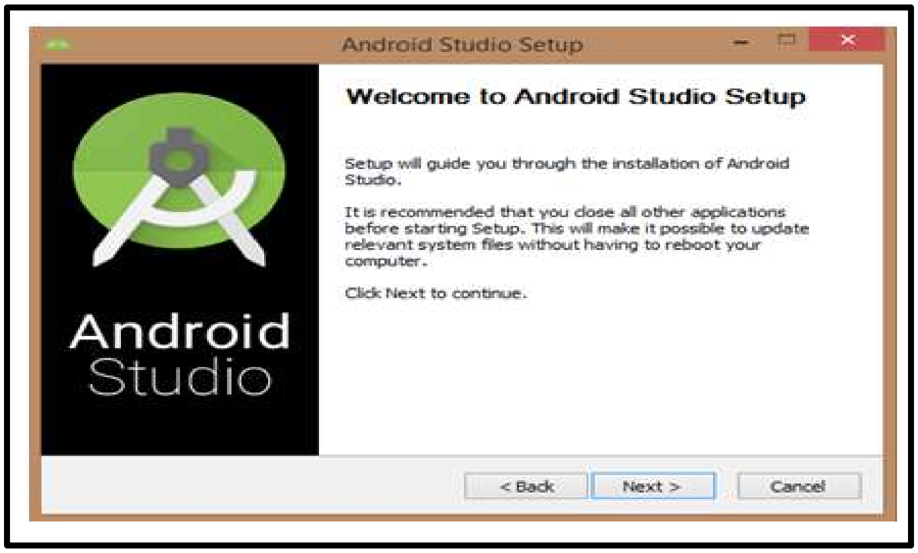
- Gradle-based build support.
- Android-specific refactoring and quick fixes.
- Lint tools to catch performance, usability, version compatibility and other problems.
- ProGuard integration and app-signing capabilities.
- Template-based wizards to create common Android designs and components.

- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations.
- Support for building Android Wear apps.
- Built-in support for Google Cloud Platform, enabling integration with Google Cloud Messaging and App Engine.

PROCEDURE:

Steps to launch Android Studio.exe are as follows: [3]

- Make sure before launching Android Studio, our Machine requires installed Java JDK. To install Java JDK, take references of Android environment setup. Then click on android studio set up as shown.



Welcome to Android Studio Setup

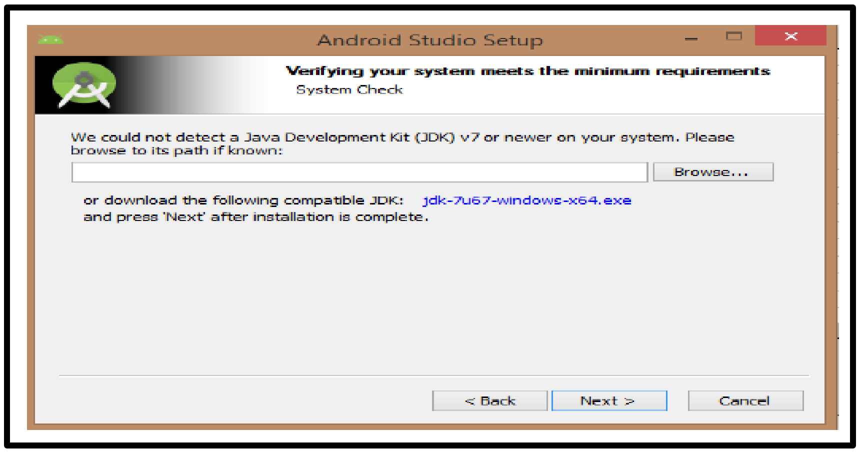
Setup will guide you through the installation of Android Studio.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Figure 2.1: Installing Android Studio

- Once you have launched Android Studio, it's time to mention JDK7 path or later version in android studio installer.



Android Studio Setup

Verifying your system meets the minimum requirements

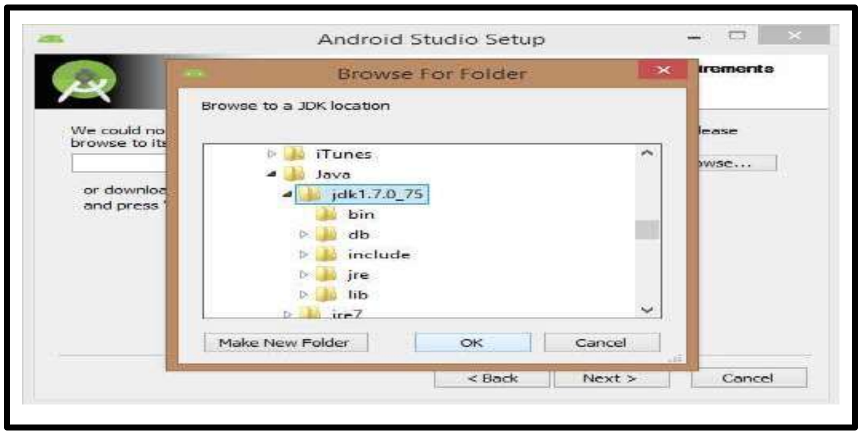
System Check

We could not detect a Java Development Kit (JDK) v7 or newer on your system. Please browse to its path if known:

or download the following compatible JDK: jdk-7u67-windows-x64.exe
and press 'Next' after installation is complete.

Figure 2.2: Mentioning JDK path

- Below the image initiating JDK to android SDK.

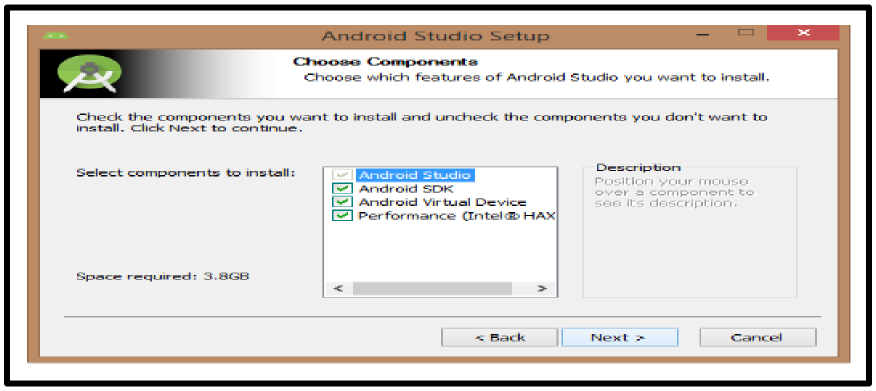


Browse For Folder

Browse to a JDK location

Figure 2.3: Initiate the JDK path after browsing

- Need to check the components, which are required to create applications, below the image have selected Android Studio, Android SDK, Android Virtual Machine and performance (Intel chip).



Android Studio Setup

Choose Components

Choose which features of Android Studio you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

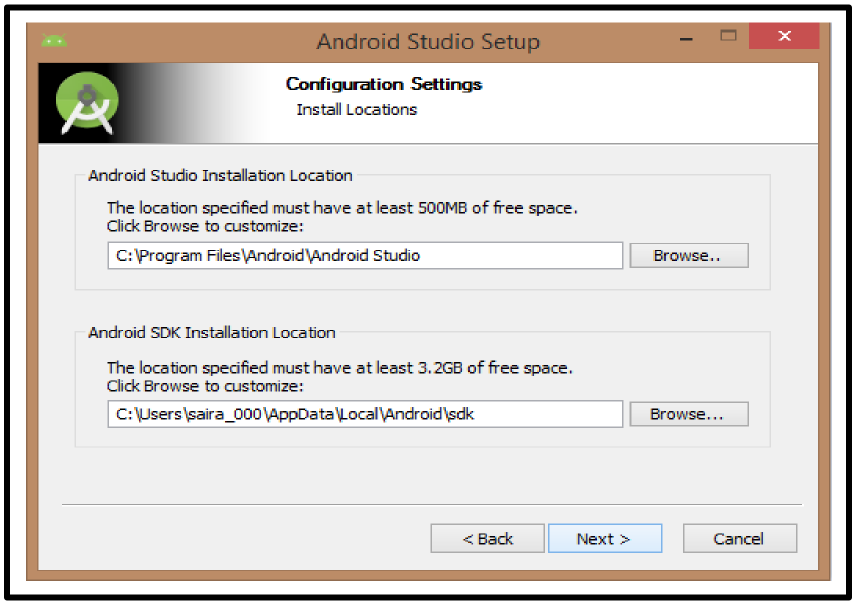
- ✓ Android Studio
- ✓ Android SDK
- ✓ Android Virtual Device
- ✓ Performance (Intel® HAX

Description
Position your mouse over a component to see its description.

Space required: 3.8GB

Figure 2.4: Choose components to install

- Need to specify the location of local machine path for Android studio and Android SDK, below the image has taken default location of windows 8.1 x64 bit architecture.



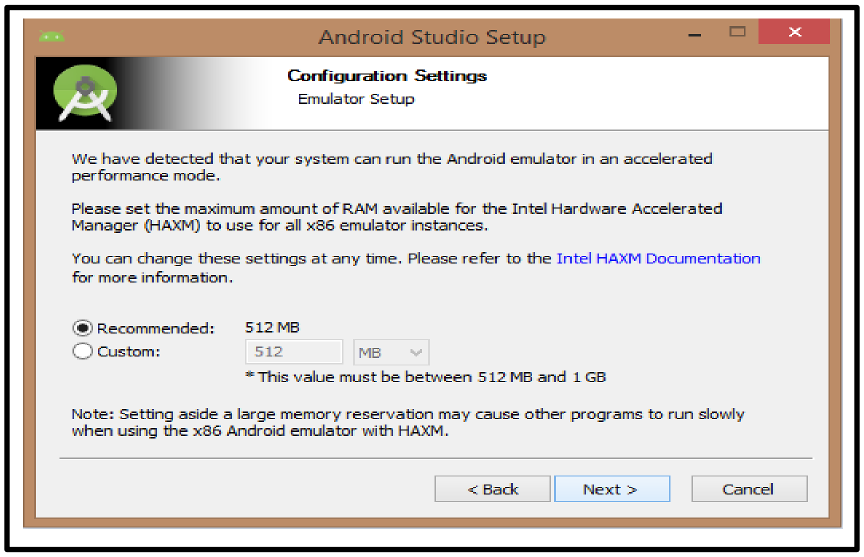
Configuration Settings
Install Locations

Android Studio Installation Location
The location specified must have at least 500MB of free space.
Click Browse to customize:
C:\Program Files\Android\Android Studio

Android SDK Installation Location
The location specified must have at least 3.2GB of free space.
Click Browse to customize:
C:\Users\saira_000\AppData\Local\Android\sdk

Figure 2.5: Browse the location for android studio and android SDK

- Need to specify the ram space for Android emulator by default it would take 512MB of local machine RAM.



Configuration Settings
Emulator Setup

We have detected that your system can run the Android emulator in an accelerated performance mode.

Please set the maximum amount of RAM available for the Intel Hardware Accelerated Manager (HAXM) to use for all x86 emulator instances.

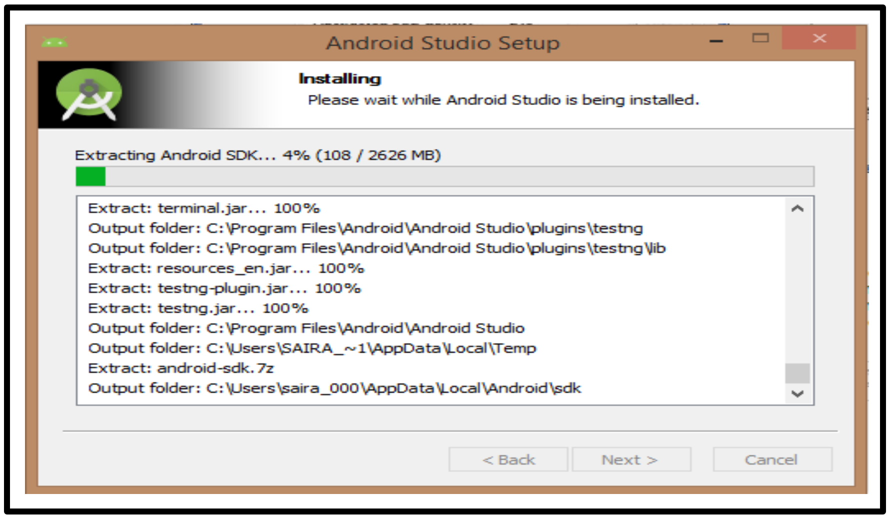
You can change these settings at any time. Please refer to the Intel HAXM Documentation for more information.

[✓ Recommended: 512 MB]
Custom: 512 MB
* This value must be between 512 MB and 1 GB

Note: Setting aside a large memory reservation may cause other programs to run slowly when using the x86 Android emulator with HAXM.

Figure 2.6: Specifying the RAM space

- At final stage, it would extract SDK packages into our local machine, it would take a while time to finish the task and would take 2626MB of Hard disk space.



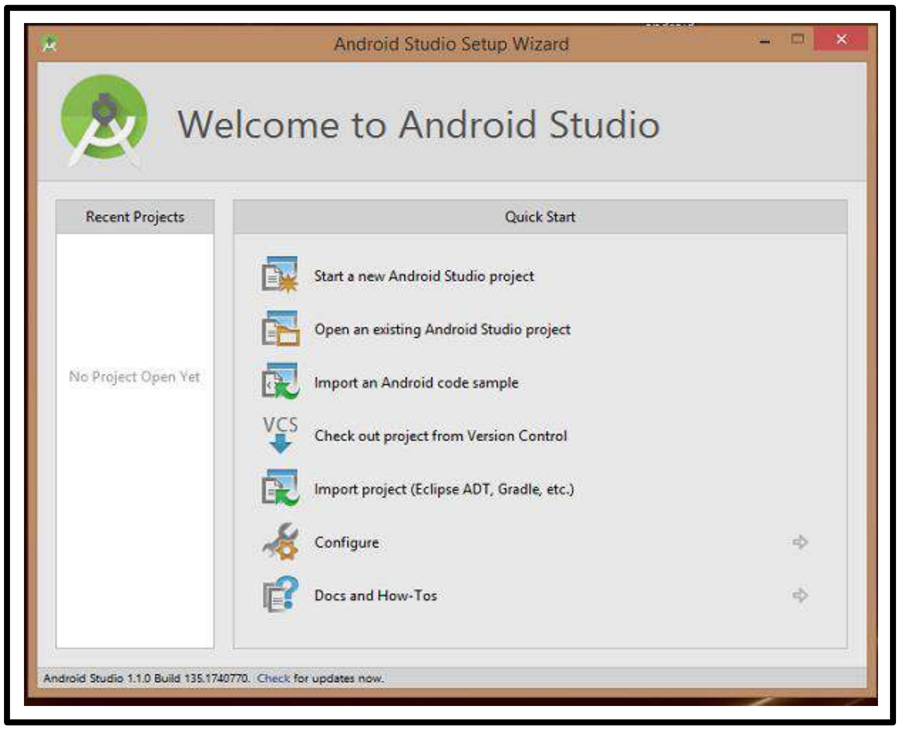
Installing

Please wait while Android Studio is being installed.

```
Extracting Android SDK... 4% (108 / 2626 MB)

Extract: terminal.jar... 100%
Output folder: C:\Program Files\Android\Android Studio\plugins\testng
Output folder: C:\Program Files\Android\Android Studio\plugins\testng\lib
Extract: resources_en.jar... 100%
Extract: testng-plugin.jar... 100%
Extract: testng.jar... 100%
Output folder: C:\Program Files\Android\Android Studio
Output folder: C:\Users\SAIRA_~1\AppData\Local\Temp
Extract: android-sdk.7z
Output folder: C:\Users\saira_000\AppData\Local\Android\sdk
```

Figure 2.7: Progress Bar of Installation process

- After done all above steps perfectly, you must get finish button and it will open android studio project with Welcome to android studio message as shown below

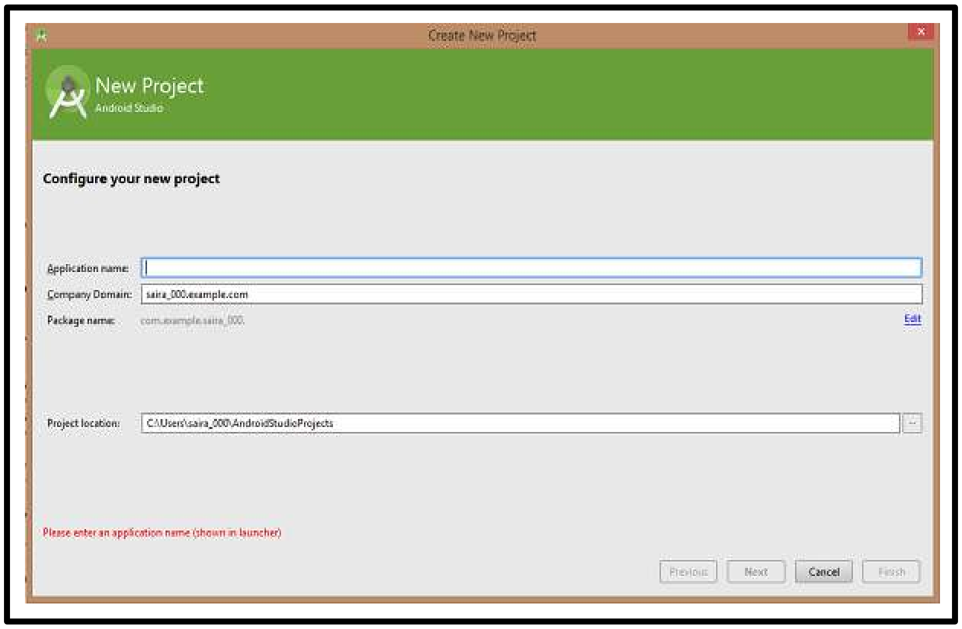


Android Studio Setup Wizard
Welcome to Android Studio
Recent Projects
No Project Open Yet
Quick Start
Start a new Android Studio project
Open an existing Android Studio project
Import an Android code sample
VCS
Check out project from Version Control
Import project (Eclipse ADT, Gradle, etc.)
Configure
Docs and How-Tos

Android Studio 1.1.0 Build 135.1740770. Check for updates now.

Figure 2.8: Android Studio set up Wizard

- You can start your application development by calling start a new android studio project. In a new installation frame should ask Application name, package information and location of the project.


Create New Project

New Project
Android Studio

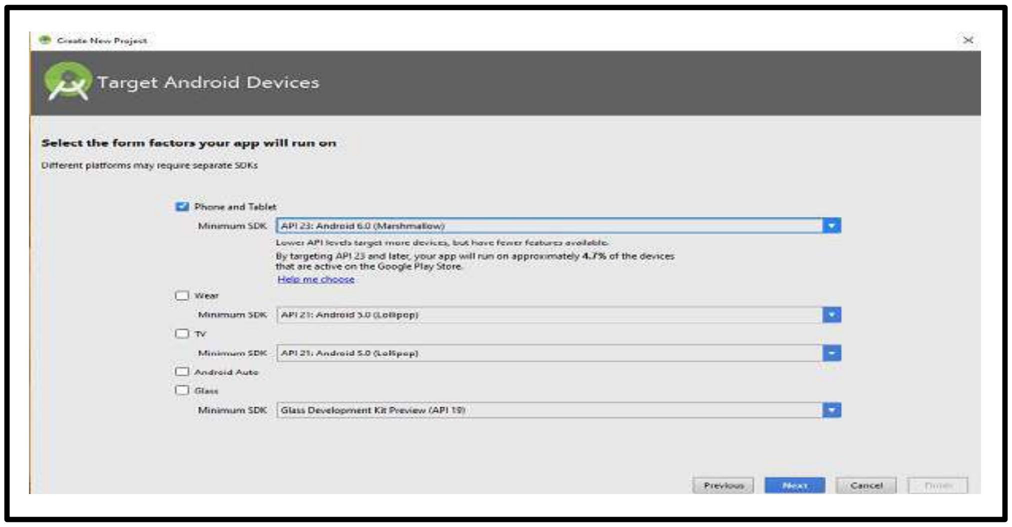
Configure your new project

Application name:
Company Domain: saira_000.example.com
Package name: com.example.saira_000.
Project location: C:\Users\saira_000\AndroidStudioProjects

Please enter an application name (shown in launcher)

Figure 2.9: Creating a new project

- After entering application name, it going to be called selects the form factors your application runs on, here need to specify Minimum SDK. I have declared as API23: Android 6.0(Marshmallow)


Create New Project

Target Android Devices

Select the form factors your app will run on
Different platforms may require separate SDKs

[✓] Phone and Tablet
Minimum SDK: API 23: Android 6.0 (Marshmallow)
Lower API levels target more devices, but have fewer features available.
By targeting API 23 and later, your app will run on approximately 4.7% of the devices that are active on the Google Play Store.
Help me choose

[ ] Wear
Minimum SDK: API 21: Android 5.0 (Lollipop)
[ ] TV
Minimum SDK: API 21: Android 5.0 (Lollipop)
[ ] Android Auto
[ ] Glass
Minimum SDK: Glass Development Kit Preview (API 19)

Figure 2.10: Selecting the form factors

- The next level of installation should contain selecting the activity to mobile, it specifies the default layout for Applications.

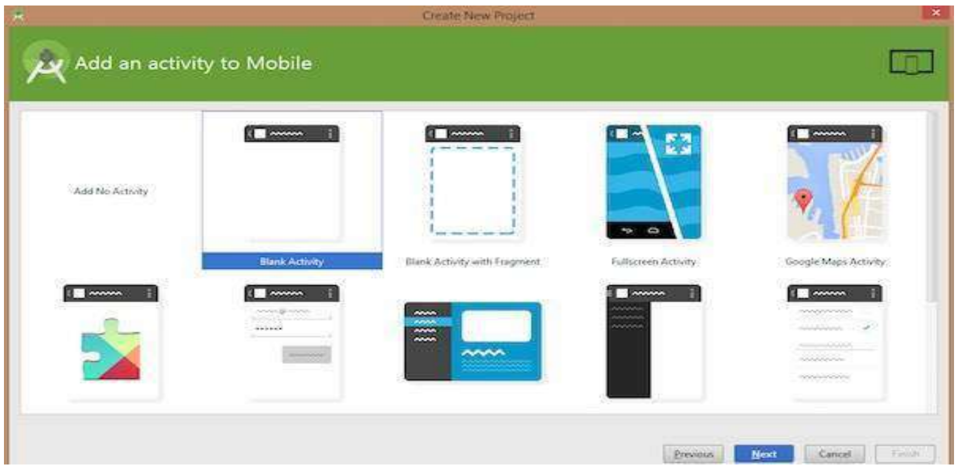


Figure 2.11: Selecting the default layout

- At the final stage it going to be open development tool to write the application code.

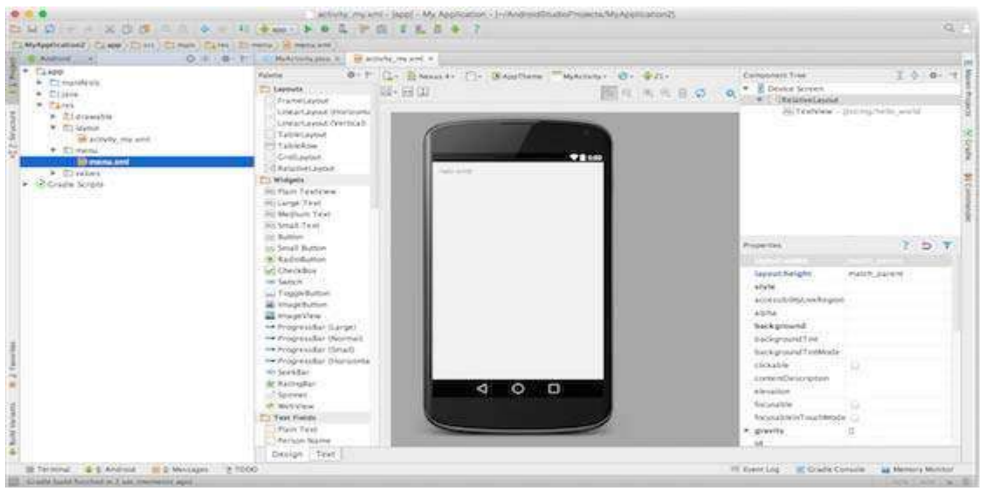


Figure 2.12: Open the .xml and .java file

Now to design Welcome application on Android Studio, further steps need to be followed:

1. Before writing the Welcome code, you must know about XML tags:

   To write the code, we need to open mainactivity.java file from app>java.

   To design and preview layout, we should redirect to App>res>layout>Activity_main.xml.

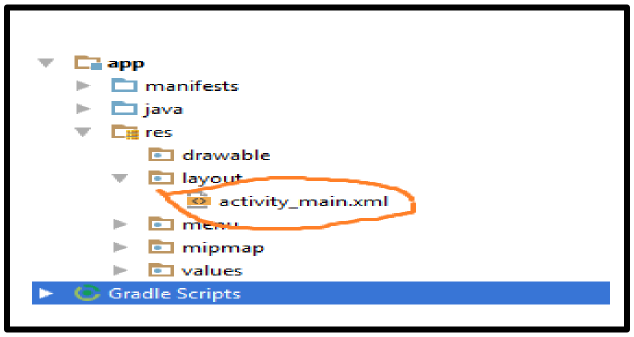


Figure 2.13: Left side pane of android studio

2. The xml code for design layout in activity_basic.xml is as shown:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_basic"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context="com.example.aaa.welcome.BasicAct">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Welcome to IGDTUW"
        android:textAlignment="center"
        android:textColor="@android:color/black"
        android:textSize="30dp"
        android:id="@+id/welcome" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Enter Your Name:"
        android:textColor="@android:color/black"
        android:id="@+id/enterName"
        android:layout_below="@+id/welcome"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:textSize="15dp"
        android:layout_marginTop="52dp" />
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="28dp"
        android:id="@+id/name"
        android:layout_below="@+id/enterName"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_marginLeft="12dp"
        android:layout_marginStart="12dp" />
    <Button
        android:text="click here"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="69dp"
        android:onClick="onClickHere"
        android:id="@+id/button"
        android:layout_below="@+id/name"
        android:layout_centerHorizontal="true" />
</RelativeLayout>
```

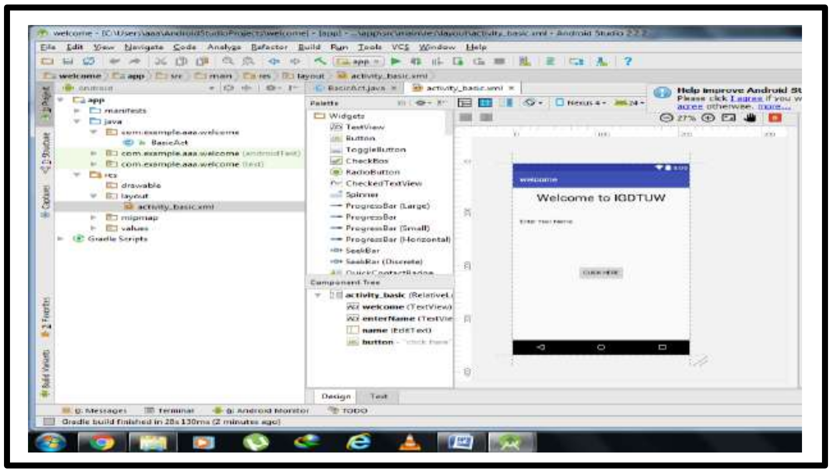


Figure 2.14: Design preview of welcome app

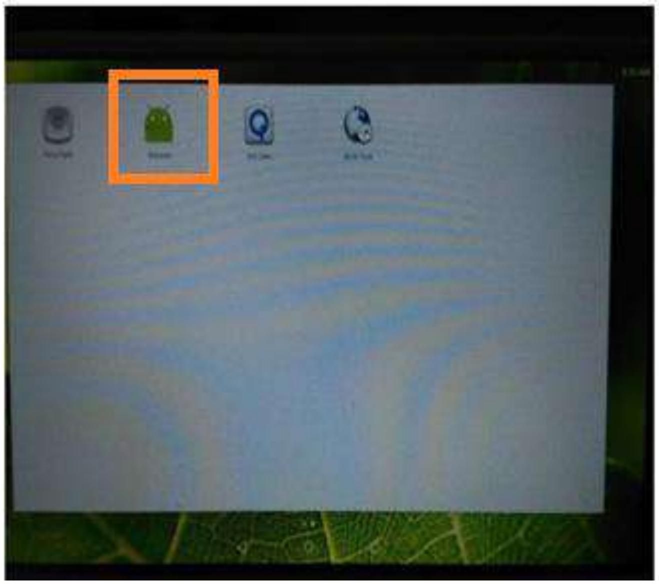
3. The code written in java file is as shown:

```
package com.example.aaa.welcome;
        import android.support.v7.app.AppCompatActivity;
        import android.os.Bundle;
        import android.view.View;
        import android.widget.EditText;
        import android.widget.Toast;
public class BasicAct extends AppCompatActivity {
    EditText editText;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_basic);
        editText = (EditText) findViewById(R.id.name);  }
    public void onClickHere(View view){
        String strName=editText.getText().toString();
        Toast.makeText(getApplicationContext(),"Hi!
"+strName,Toast.LENGTH_LONG).show();
    }
}
```

4. Now, the application is ready to be run on any android device. To port the application in the snapdragon board, first we have to boot the snapdragon and connect to the TFT screen.
5. Next, connect the snapdragon board to the laptop via a Micro USB cable.
6. On the Android Studio start the compilation of the project. If compilation is successful, the APK file of the project would get transferred to the Snapdragon and the application will get displayed on the TFT screen automatically.
7. There will be Welcome app icon on the menu of TFT screen and after clicking on that icon Welcome app will open.
8. After opening it will ask "Enter your name:" and then virtual keyboard will appear. After entering name, a pop up window will appear saying "Hi!" along with the name.

OUTPUT SCREENSHOTS:

Some screenshots of output are as shown below.



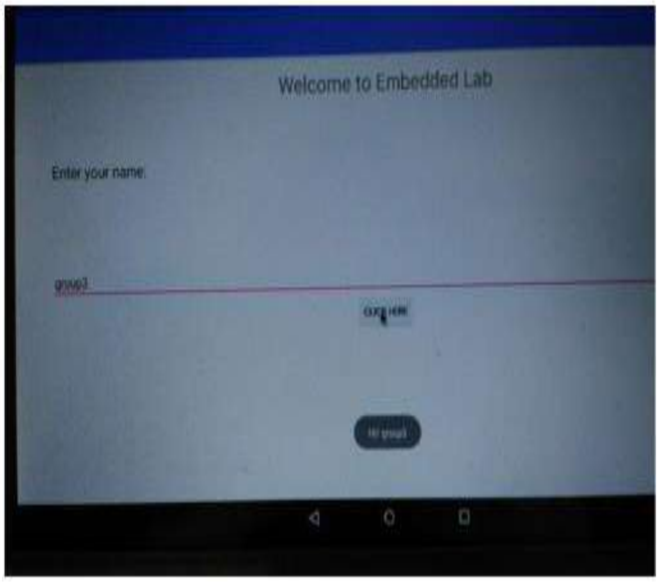


Figure 2.15: Welcome App icon on the menu

Figure 2.16: Launching the Application on Snapdragon

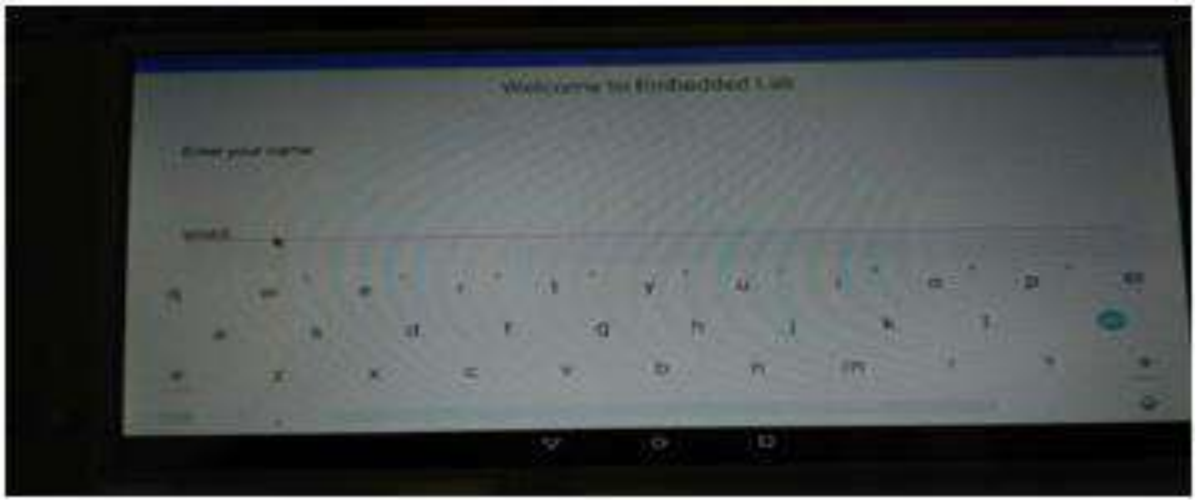


Figure 2.17: Entering the text

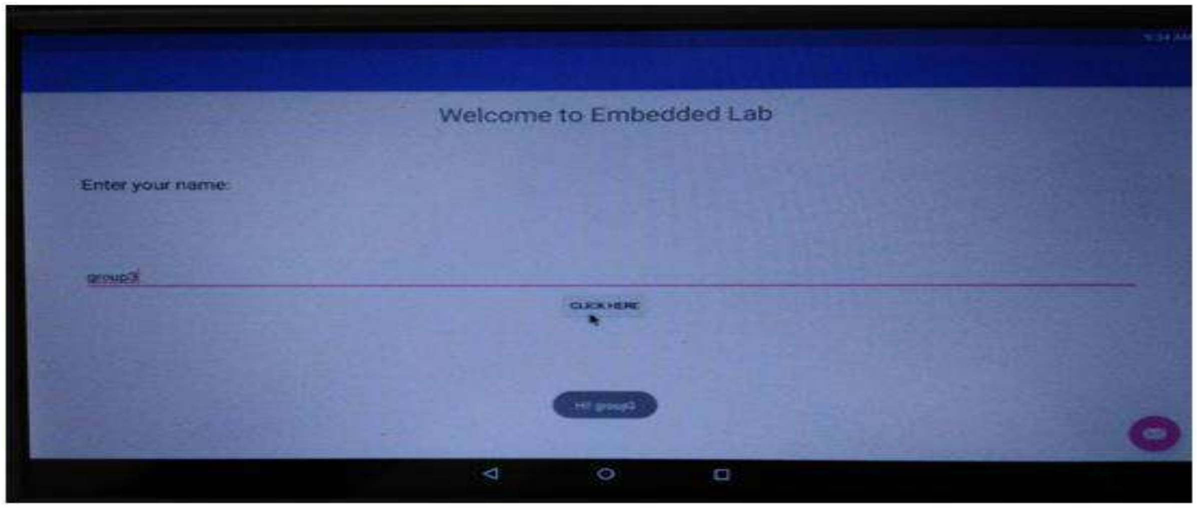


Figure 2.18: Pop up appears

RESULT: We have successfully installed Android Studio and deployed the Welcome app on the Snapdragon Dragon Board 410c.

EXPERIMENT-3

AIM: To design an application to enable and disable the communication modules in the Snapdragon Dragon Board 410c.

COMPONENTS REQUIRED:

Hardware Components: Snapdragon Dragon Board 410c, HDMI-to HDMI cable, 2 AC adapters (12 V), TFT screen, Driver Board for TFT screen. Mouse, Micro USB cable, Laptop/PC

Software Components: Android Studio or any other such IDE installed on the PC/Laptop.

THEORY:

Bluetooth

Bluetooth is a wireless technology standard for exchanging data over short distances (using short-wavelength UHF radio waves in the ISM band from 2.4 to 2.485 GHz) from fixed and mobile devices, and building personal area networks (PANs). Invented by telecom vendor Ericsson in 1994, it was originally conceived as a wireless alternative to RS-232 data cables. It can connect several devices, overcoming problems of synchronization.

Bluetooth is managed by the Bluetooth Special Interest Group (SIG), which has more than 25,000 member companies in the areas of telecommunication, computing, networking, and consumer electronics. The IEEE standardized Bluetooth as IEEE 802.15.1, but no longer maintains the standard.

The Dragon Board has Bluetooth 4.1, integrated digital core.

WiFi

WIFI is a technology that allows electronic devices to connect to a wireless LAN (WLAN), mainly using the 2.4 gigahertz (12 cm) UHF and 5 gigahertz (6 cm) SHF ISM radio bands. A WLAN is usually password protected, but may be open, which allows any device within its range to access the resources of the WLAN network.

The Wi-Fi Alliance defines Wi-Fi as any "wireless local area network" (WLAN) product based on the Institute of Electrical and Electronics Engineers' (IEEE) 802.11 standards."Wi-Fi" is a trademark of the Wi-Fi Alliance.

The Dragon Board has Wi-Fi 802.11 b/g/n 2.4GHz, integrated digital core.

PROCEDURE:

1. Create a new project on Android Studio in the Laptop/PC.
2. Name the project as Communication Modules or any other suitable name.
3. In the MainActivity.java file include the following code:

```
package com.hfad.communication_modules_demo;

import android.app.Activity;

import android.bluetooth.BluetoothAdapter;

import android.content.Context;

import android.content.Intent;

import android.net.wifi.WifiManager;

import android.os.Bundle;

import android.widget.CompoundButton;

import android.widget.Switch;

import android.widget.TextView;

public class MainActivity extends Activity {

   Switch switchButton;

   TextView textview, textViewWifi;

   BluetoothAdapter bluetoothadapter;
```

```
    int i = 1;

    Intent bluetoothIntent;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        switchButton = (Switch)findViewById(R.id.switch1);

        textview = (TextView)findViewById(R.id.textView1);

        textViewWifi = (TextView)findViewById(R.id.textView2);

        bluetoothadapter = BluetoothAdapter.getDefaultAdapter();

      switchButton.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {

            @Override

            public void onCheckedChanged(CompoundButton buttonView,
boolean isChecked) {

                if(isChecked)

                {

                    BluetoothEnable();

                }

                else {

                    BluetoothDisable();
                }
            }
        });
        Switch toggle = (Switch) findViewById(R.id.wifi_switch);
        toggle.setOnCheckedChangeListener(new
    CompoundButton.OnCheckedChangeListener() {
            public void onCheckedChanged(CompoundButton buttonView,
    boolean isChecked) {
                if (isChecked) {
                    toggleWiFi(true);
                    textViewWifi.setText("Wifi ON");
                } else {
                    toggleWiFi(false);
                    textViewWifi.setText("Wifi OFF");
                }
            }
        });
    }
    public void BluetoothEnable(){
        bluetoothIntent=new
    Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(bluetoothIntent, i);
```

```
            textview.setText("Bluetooth ON");
        }
        public void BluetoothDisable(){
            bluetoothadapter.disable();
            textview.setText("Bluetooth OFF");
        }
        public void toggleWiFi(boolean status) {
            WifiManager wifiManager = (WifiManager) this
                    .getSystemService(Context.WIFI_SERVICE);
            if (status == true && !wifiManager.isWifiEnabled()) {
                wifiManager.setWifiEnabled(true);
            } else if (status == false && wifiManager.isWifiEnabled()) {
                wifiManager.setWifiEnabled(false);
            }
        }
    }
```

4. Now we define the activity_main.xml. In this project we require 2 switches to toggle the bluetooth and wifi ON and OFF. We also use 2 TextView to display the bluetooth/ wifi status.

   To define a switch and TextView we use the following syntax in activity_main.xml file:

```
<TextView
   android:id="@+id/textView2"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_centerHorizontal="true"
   android:layout_marginBottom="62dp"
   android:gravity="center"
   android:text="Wifi status show here"
   android:layout_below="@id/switch1"
   android:textAppearance="?android:attr/textAppearanceLarge" />
<Switch
   android:id="@+id/wifi_switch"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:checked="false"
   android:layout_centerHorizontal="true"
   android:layout_centerVertical="true"
   android:layout_below="@id/textView2"/>
```

5. Now, the application is ready to be run on any android device. To port the application in the snapdragon board, first we have to boot the snapdragon and connect to the TFT screen.

6. Next, connect the snapdragon board to the laptop via a Micro USB cable.

7. On the Android Studio start the compilation of the project. If compilation is successful, the APK file of the project would get transferred to the Snapdragon and the application will get displayed on the TFT screen automatically.

OUTPUT SCREENSHOTS:

App icon on the Menu screen:

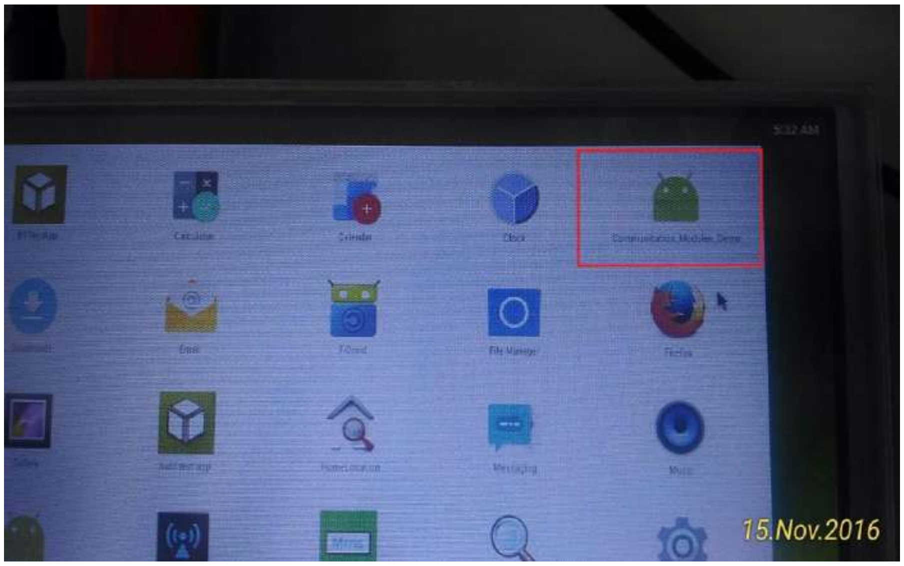


Figure 3.1: Communication Modules Demo Application

Launching the application:

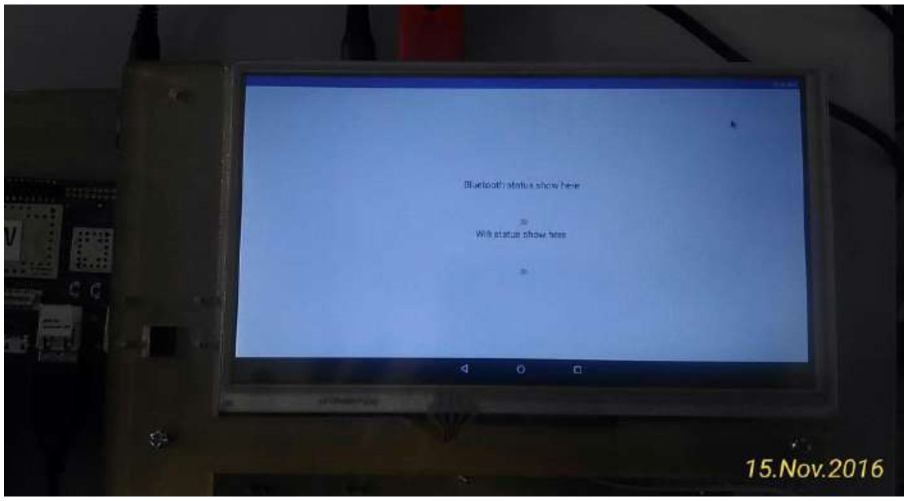


Figure 3.2: Launching the application for the first time

LEDs on the Board to indicate the WiFi and Bluetooth status:

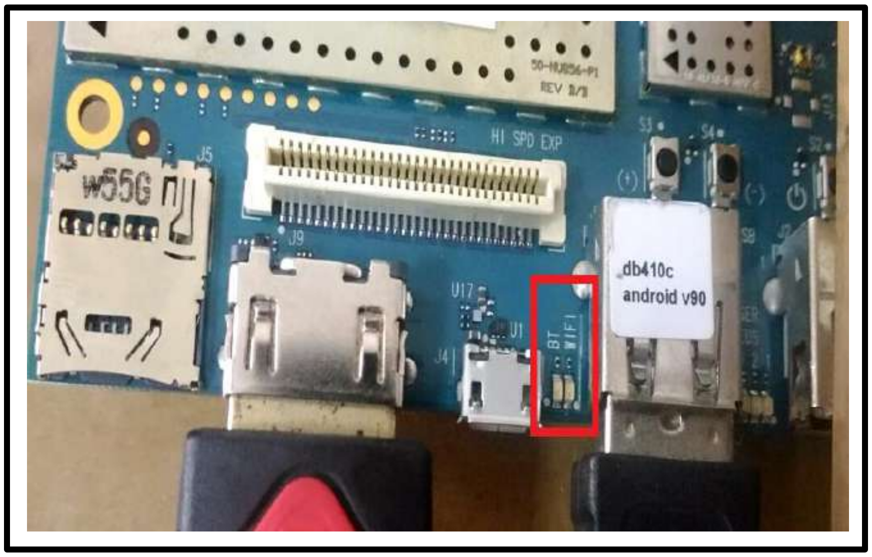


Figure 3.3: The WiFi and Bluetooth LEDs

Enabling Wi-Fi: The yellow LED is ON the Board signifying that the Bluetooth is ON.

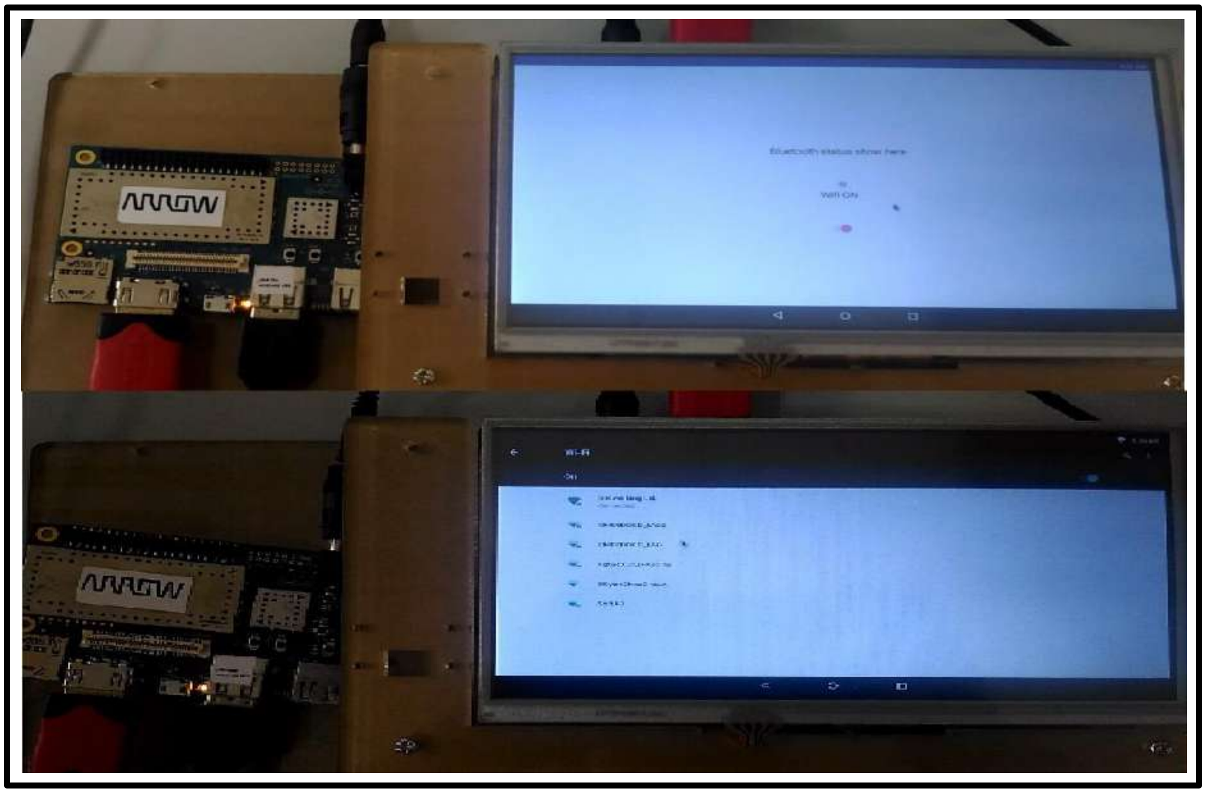


Figure 3.4: WiFi is ON and is connected to an available network

Enabling Bluetooth in Snapdragon:

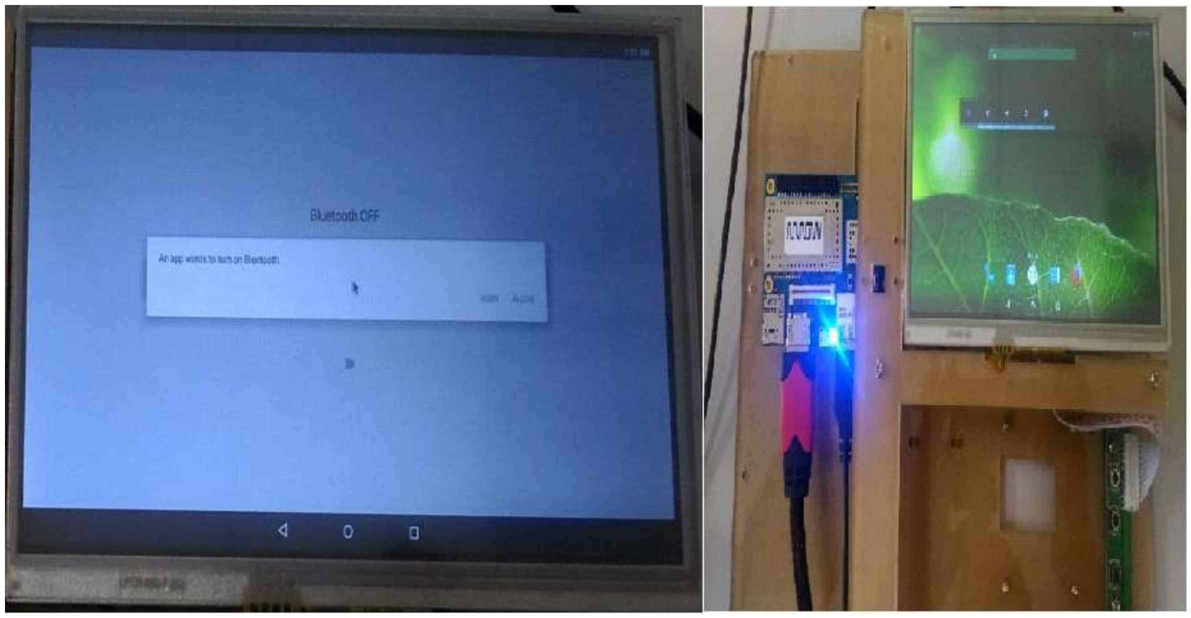


Fig3.5: The Bluetooth is enabled and the LED glows in blue color signifying the Bluetooth is enabled

RESULT: We have successfully enabled the in-build Bluetooth and WiFi module via the android application.

EXPERIMENT-4

AIM: To design an application to perform basic calculations on Snapdragon board 410c.

COMPONENTS REQUIRED:

Hardware Components: Dragon Board 410c, HDMI to HDMI cable, 2 AC Adapters (12 V), TFT Screen, Driver Board for TFT Screen, Micro USB cable, Mouse.

Software Components: Android Studio

THEORY:

1.) **Dragon Board 410c:** The Dragon Board 410c is the first development board based on a Qualcomm Snapdragon 400 series processor. It features advanced processing power, Wi-Fi, Bluetooth connectivity, and GPS, all packed into a board the size of a credit card. Based on the 64-bit capable Snapdragon 410E processor, the Dragon Board 410c is designed to support rapid software development, education and prototyping.

2.) **Android Studio:** Android Studio is the official integrated development environment (IDE) for Android platform development. It was announced on May 16, 2013 at the Google I/O conference. Android Studio is freely available under the Apache License 2.0.Android Studio was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0. Based on JetBrains' IntelliJ IDEA software, Android Studio is designed specifically for Android development. It is available for download on Windows, Mac OS X and Linux, and replaced Eclipse Android Development Tools (ADT) as Google's primary IDE for native Android application development.

PROCEDURE:

1) Create a new android project in Android Studio.
2) Name the project as My Basic Calculator or any other suitable name.
3) In the MainActivity.Java file include the following code:

```
package com.example.dell.mybasiccalculator;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

public class MainActivity extends AppCompatActivity implements
View.OnClickListener{

  private Button btnAdd,btnSub,btnDivide,btnMul;

  private TextView tvresult;

  private EditText etfirst,etsecond;

  @Override

  protected void onCreate(Bundle savedInstanceState) {

      super.onCreate(savedInstanceState);

      setContentView(R.layout.activity_main);

      init();

  }

  private void init() {

      btnAdd= (Button)findViewById(R.id.btnAdd);
```

```
    btnSub= (Button)findViewById(R.id.btnSubtract);

    btnDivide= (Button)findViewById(R.id.btnDivide);

    btnMul= (Button)findViewById(R.id.btnMultiply);

    etfirst= (EditText)findViewById(R.id.etFirstNumber);

    etsecond= (EditText)findViewById(R.id.etSecondNumber);

    tvresult= (TextView)findViewById(R.id.tvResult);

    btnAdd.setOnClickListener(this);

    btnSub.setOnClickListener(this);

    btnDivide.setOnClickListener(this);

    btnMul.setOnClickListener(this);

}

@Override

public void onClick(View view) {

    String num1= etfirst.getText().toString();

    String num2= etsecond.getText().toString();

    switch(view.getId()){

        case R.id.btnAdd:

            Double addition= Double.parseDouble(num1) + Double.parseDouble(num2);

            tvresult.setText(String.valueOf(addition));

            break;

        case R.id.btnSubtract:
            Double subtraction= Double.parseDouble(num1) - Double.parseDouble(num2);

            tvresult.setText(String.valueOf(subtraction));

            break;

        case R.id.btnDivide:

            try {

                Double division = Double.parseDouble(num1) / Double.parseDouble(num2);

                tvresult.setText(String.valueOf(division));

            }catch(Exception e){

                tvresult.setText("cannot DIVIDE");

            }

            break;

        case R.id.btnMultiply:

            Double multiply = Double.parseDouble(num1) * Integer.parseInt(num2);

            tvresult.setText(String.valueOf(multiply));

            break;

}}}
```

4) Now set the layout of the application in the .xml file. As follows:

   We have used 4 TextViews(one for displaying the heading,2 for displaying the first number and second number, last one to display the result), 2 EditText to enter the 1st and 2nd parameter and we have used 4 buttons(Addition, Subtraction, Multiplication, Division).

To define TextView, EditText and Buttons use following logic:

```
<EditText
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:inputType="numberDecimal"
  android:ems="10"
  android:layout_below="@+id/editText"
  android:layout_alignRight="@+id/editText"
  android:layout_alignEnd="@+id/editText"
  android:id="@+id/etFirstNumber" />
<TextView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:inputType="textPersonName"
  android:text="Second Number"
  android:ems="10"
  android:layout_below="@+id/etFirstNumber"
  android:layout_alignRight="@+id/etFirstNumber"
  android:layout_alignEnd="@+id/etFirstNumber"
  android:id="@+id/editText3" />
```

```
<Button
  android:text="+"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:id="@+id/btnAdd" />
```

5) Now we can run the application on any Android Device. So to transfer the application to the dragon board 410c we need to connect it to laptop via micro USB cable and if there are no errors in the code the application will run successfully on the dragon board 410c.

SCREENSHOTS:

The icon of the app that we ported onto the dragon board 410c.

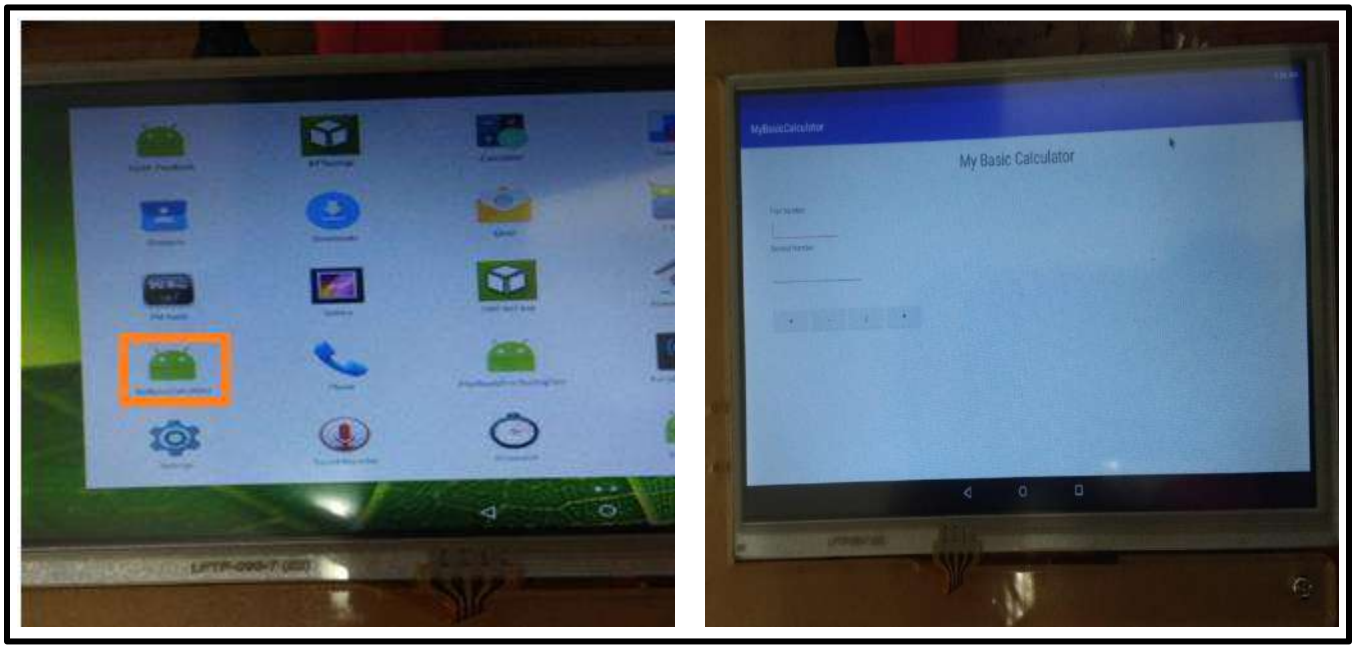


Figure 4.1: Launching the application on the snapdragon

Displaying the virtual keyboard and input given to application:

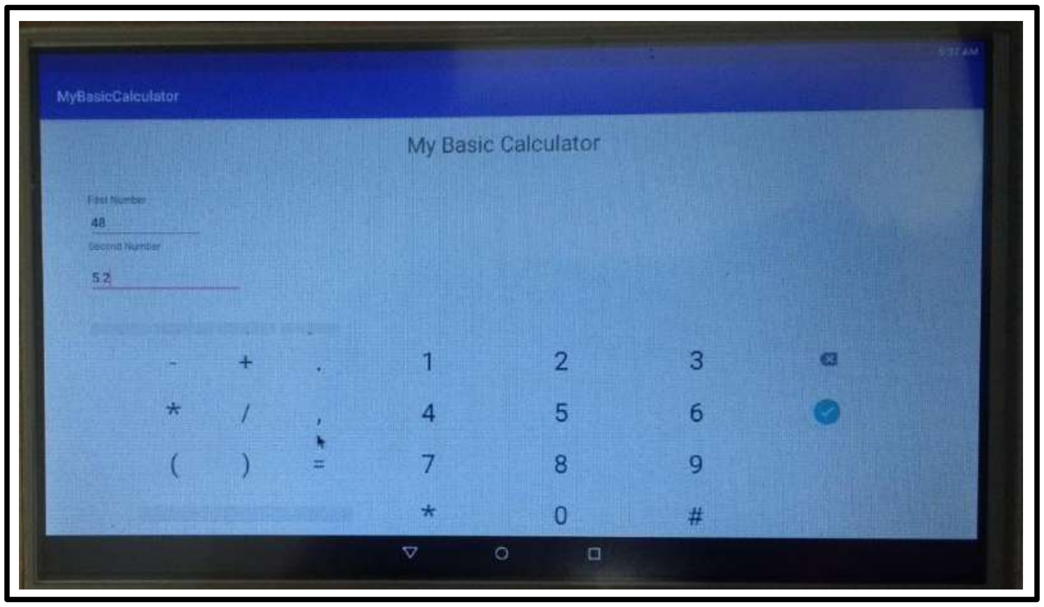


Figure 4.2: Displaying Virtual keyboard

Displaying the result after performing the desired calculation (addition in this case):

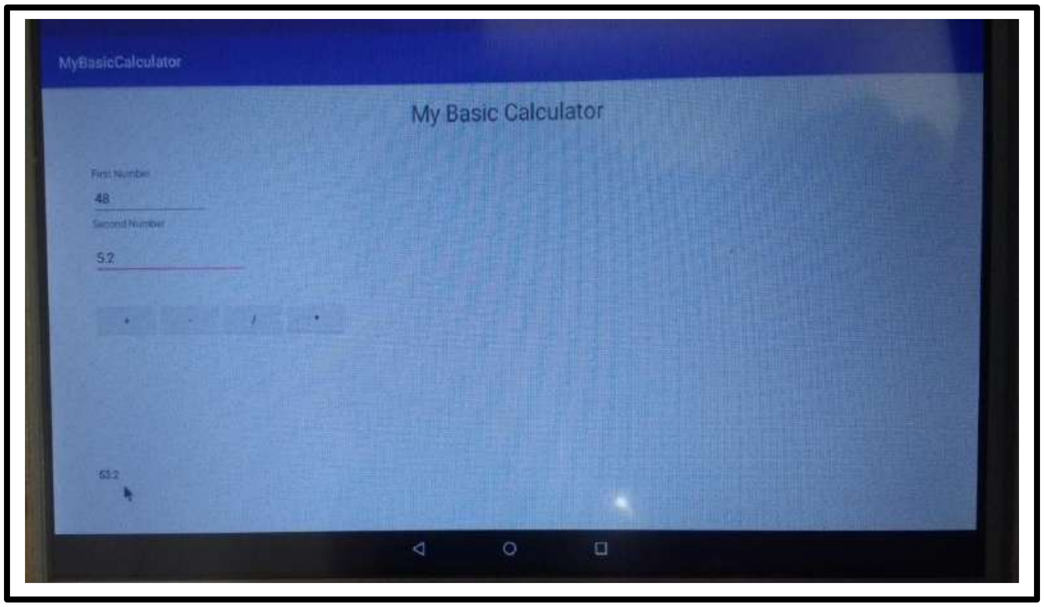


Figure 4.3: Displaying addition results

Displaying the result in after performing calculation (Multiplication in this case):

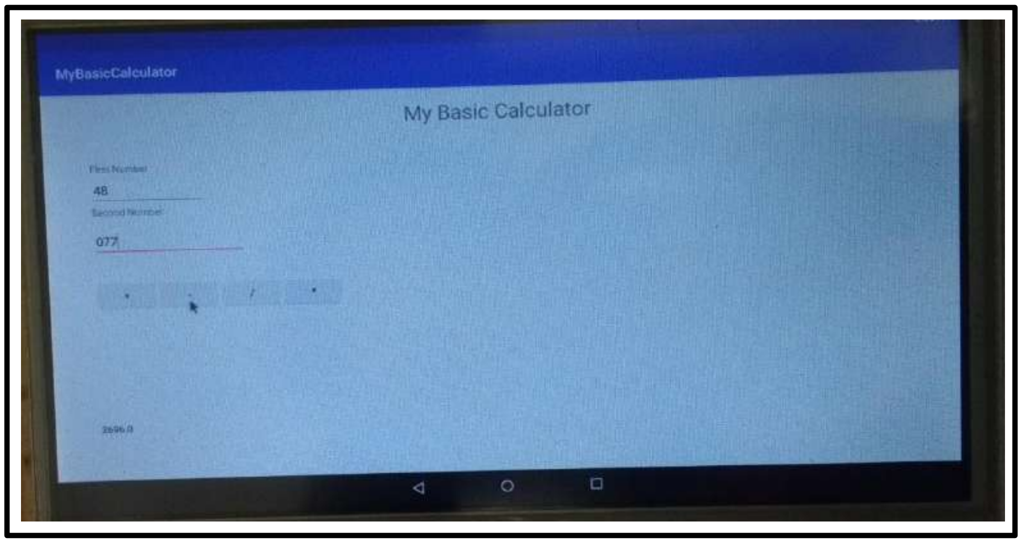


Figure 4.4: Displaying multiplication results

<u>RESULT:</u> We have successfully deployed the application on the Dragon Board 410c.

EXPERIMENT NO-5

AIM: To design an image viewer app and deploy it on dragon board 410c.

COMPONENTS REQUIRED:

Hardware Components: Dragon Board 410c, HDMI to HDMI cable, 2 AC Adapters (12 V), TFT Screen, Driver Board for TFT Screen, Micro USB cable, mouse.

Software Components: Android Studio or any other such IDE installed on the PC/Laptop.

THEORY:

An image viewer or image browser is a computer program that can display stored graphical images; it can often handle various graphics file formats. Such software usually renders the image according to properties of the display such as color depth, display resolution, and color profile.

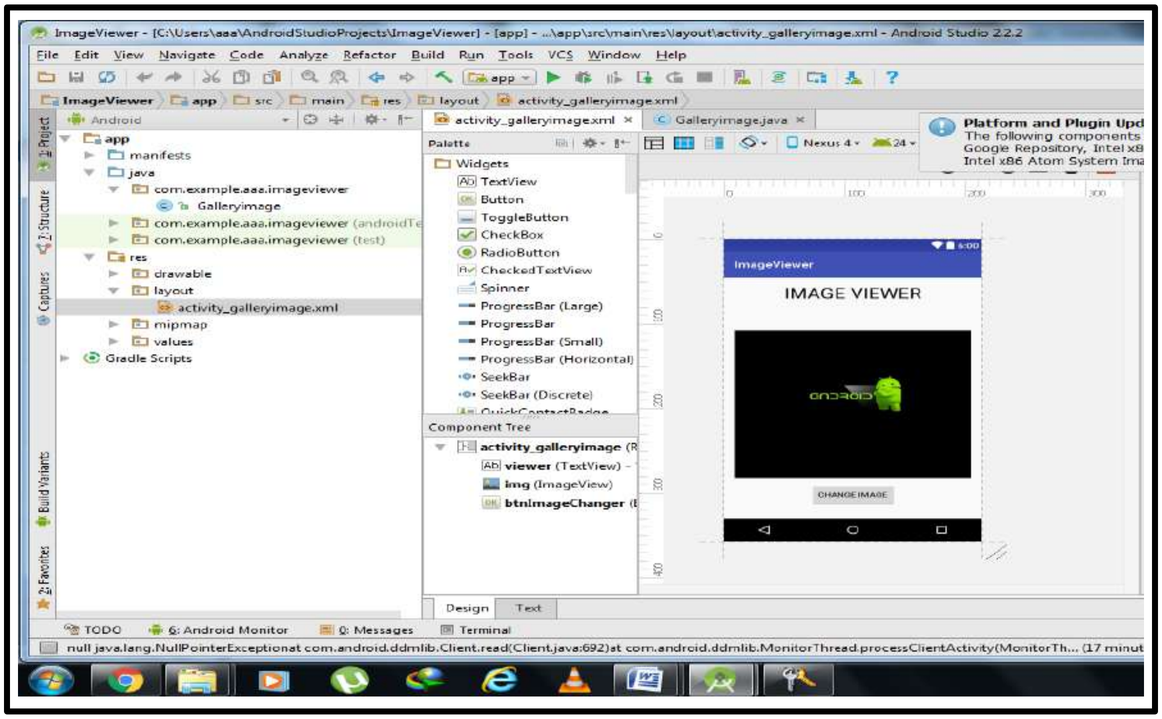
PROCEDURE:

1. Create a new project on Android Studio in the Laptop/PC.
2. Name the project as “IMAGE VIEWER” or any other suitable name.
3. In the Galleryimage.java file include the following code:

```
package com.example.aaa.imageviewer;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;

public class Galleryimage extends AppCompatActivity {
    private static ImageView imageView;
    private static Button buttonSh;

    private int current_image_index;
    int[] images={R.drawable.photo,R.drawable.ic,R.drawable.picy};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_galleryimage);
        buttonClick();
    }

    public void buttonClick(){
        imageView=(ImageView)findViewById(R.id.img);
        buttonSh=(Button)findViewById(R.id.btnImageChanger);
        buttonSh.setOnClickListener(
                new View.OnClickListener() {
                    public void onClick(View v) {
                        current_image_index++;
                        current_image_index = current_image_index % images.length;
                        imageView.setImageResource(images[current_image_index]);
                    }
                }
        );
    }
}
```

4. Now we define the activity_galleryimage.xml include the following code:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_galleryimage"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context="com.example.aaa.imageviewer.Galleryimage">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="IMAGE VIEWER"
        android:textAlignment="center"
        android:textSize="30dp"
        android:textColor="@android:color/black"
        android:id="@+id/viewer"
        />

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@+id/viewer"
        android:layout_alignParentStart="true"
        android:layout_marginTop="40dp"
        android:layout_marginBottom="50dp"
        android:src="@drawable/ic"
        android:id="@+id/img" />
    <Button
        android:text="Change Image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="10dp"
        android:id="@+id/btnImageChanger"
        />
</RelativeLayout>
```



Platform and Plugin Upd
The following components
Google Repository, Intel x8
Intel x86 Atom System Ima

null java.lang.NullPointerExceptionat com.android.ddmlib.Client.read(Client.java:692)at com.android.ddmlib.MonitorThread.processClientActivity(MonitorTh... (17 minut

Figure 5.1: Design Wizard of Image Viewer

5. Now, the application is ready to be run on any android device. To port the application in the snapdragon board, first we have to boot the snapdragon and connect to the TFT screen.
6. Next, connect the snapdragon board to the laptop via a Micro USB cable.
7. On the Android Studio start the compilation of the project. If compilation is successful, the APK file of the project would get transferred to the Snapdragon and the application will get displayed on the TFT screen automatically.
8. Now, as we click on the "CHANGE IMAGE", the image viewer will change the images and will show different images as stored in database.

OUTPUT SCREENSHOTS:

The icon of the app that we ported onto the dragon board 410c.

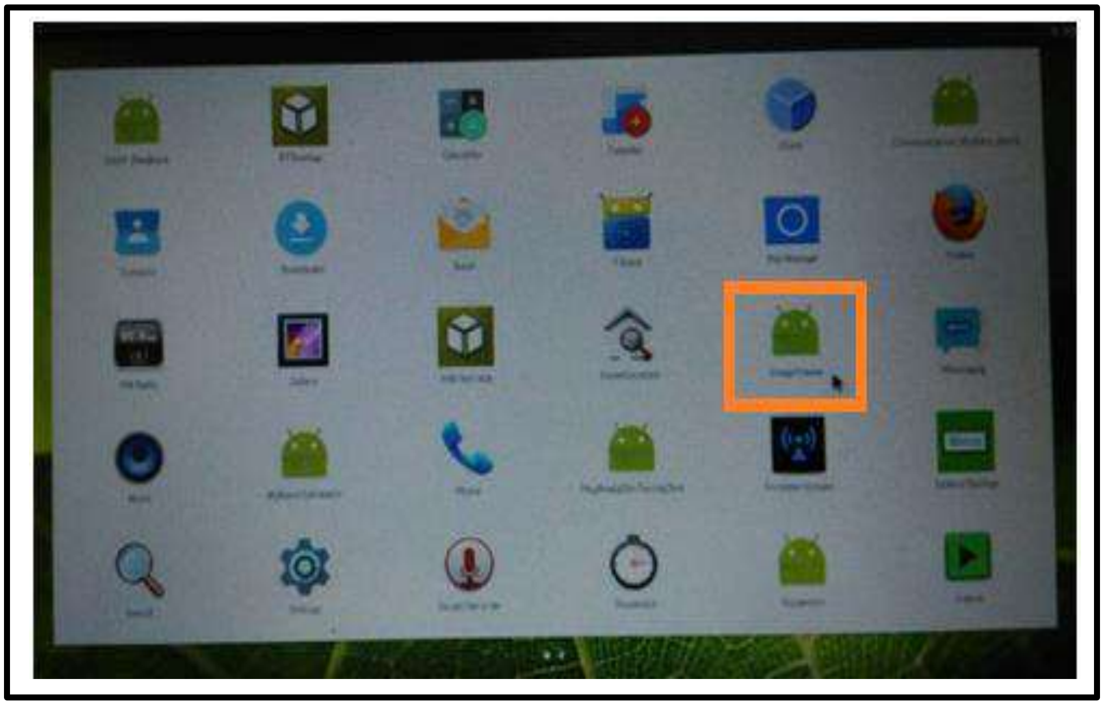


Figure 5.1: Image viewer icon launched on Snapdragon

Launching the application on the snapdragon

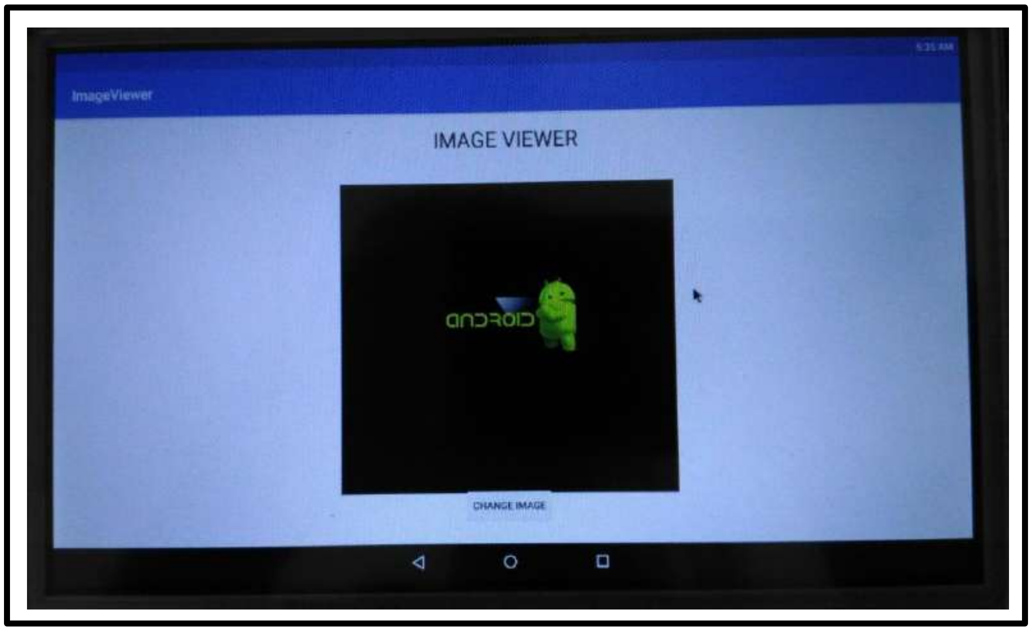


Figure 5.2: Displaying images using image viewer

<u>RESULT:</u> We have successfully deployed the image viewer app and displayed images on dragon board 410c.

EXPERIMENT-6

AIM: To design an application of stopwatch with features of Start, Reset and Stop. Port the application on the Snapdragon DragonBoard 410c.

COMPONENTS REQUIRED:

Hardware Components: Snapdragon DragonBoard 410c, HDMI to HDMI cable, 2 AC adapters (12 V), TFT screen, Driver Board for TFT screen. Mouse, Micro USB cable, Laptop/PC

Software Components: Android Studio or any other such IDE installed on the PC/Laptop.

THEORY:

A stopwatch is a handheld timepiece designed to measure the amount of time elapsed from a particular time when it is activated to the time when the piece is deactivated. A large digital version of a stopwatch designed for viewing at a distance, as in a sports stadium, is called a stop clock. In manual timing, the clock is started and stopped by a person pressing a button. In fully automatic time, both starting and stopping are triggered automatically, by sensors.

PROCEDURE:

1. Create a new project on Android Studio in the Laptop/PC.
2. Name the project as Stop Watch or any other suitable name.
3. In the MainActivity.java file include the following code:

```
package com.hfad.stopwatch;

import android.app.Activity;

import android.view.View;

import android.os.Bundle;

import android.os.Handler;
import android.widget.TextView;
public class StopwatchActivity extends Activity{
  private int seconds = 0;
  private boolean running;
  private boolean wasRunning;
  protected void onCreate(Bundle savedInstanceState){
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_stopwatch);
      if(savedInstanceState != null){
          seconds = savedInstanceState.getInt("seconds");
          running = savedInstanceState.getBoolean("running");
          wasRunning = savedInstanceState.getBoolean("wasRunning");
      }
      runTimer();
  }
  @Override
  public void onSaveInstanceState(Bundle savedInstanceState){
      savedInstanceState.putInt("seconds", seconds);
      savedInstanceState.putBoolean("running", running);
      savedInstanceState.putBoolean("wasRunning", wasRunning);
  }
```

```
@Override
protected void onStop(){
    super.onStop();
    wasRunning = running;
    running = false;  }
@Override
protected void onStart(){
    super.onStart();
    if(wasRunning){
        running = true;}
}
public void onClickStart(View view){
    running = true;   }
public void onClickStop(View view){
    running = false; }
public void onClickReset(View view){
    running = false;
    seconds = 0; }
private void runTimer() {
    final TextView timeView = (TextView)findViewById(R.id.time_view);
```

```
        final Handler handler = new Handler();

        handler.post(new Runnable() {

            @Override

            public void run() {

                int hours = seconds / 3600;

                int minutes = (seconds % 3600) / 60;

                int secs = seconds % 60;

                String time = String.format("%d:%02d:%02d", hours, minutes,
secs);

                timeView.setText(time);

                if(running){

                    seconds++;

                }

                handler.postDelayed(this, 1000);

            }

        });

    }

}
```

4. Now we define the activity_main.xml. In this project we require 3 buttons for implementing functions of Start, Stop and Reset the timer. We also use a TextView to display the running timer.

   To define a Button and TextView we use the following syntax:

```
<TextView
   android:id="@+id/time_view"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignParentTop="true"
   android:layout_centerHorizontal="true"
   android:layout_marginTop="0dp"
   android:text=""
   android:textAppearance="?android:attr/textAppearanceLarge"
   android:textSize="92sp" />
<Button
   android:id="@+id/start_button"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_below="@+id/time_view"
   android:layout_centerHorizontal="true"
   android:layout_marginTop="20dp"
   android:onClick="onClickStart"
   android:text="@string/start"/>
```

5. Now, the application is ready to be run on any android device. To port the application in the snapdragon board, first we have to boot the snapdragon and connect to the TFT screen.

6. Next, connect the snapdragon board to the laptop via a Micro USB cable.
7. On the Android Studio start the compilation of the project. If compilation is successful, the APK file of the project would get transferred to the Snapdragon and the application will get displayed on the TFT screen automatically.

<u>OUTPUT SCREENSHOTS:</u>

Displaying the Stop Watch App icon on the menu:

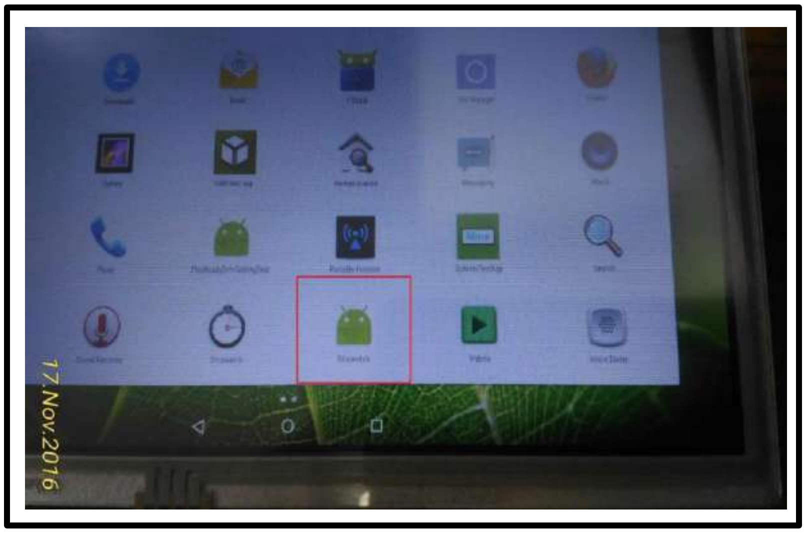


Figure 6.1: Displaying Stop Watch icon

Launching the Application on Snapdragon as shown on TFT screen below:

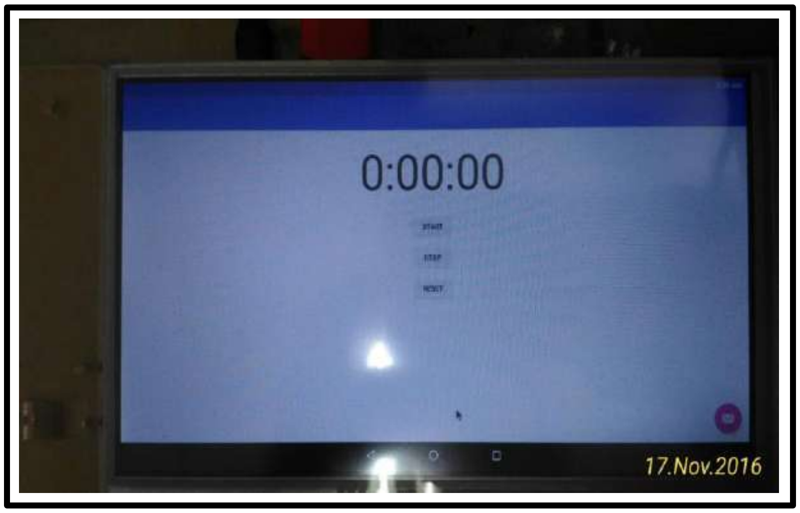


Figure 6.2: Launching the app

Starting the Timer: The timer starts on clicking the Start button

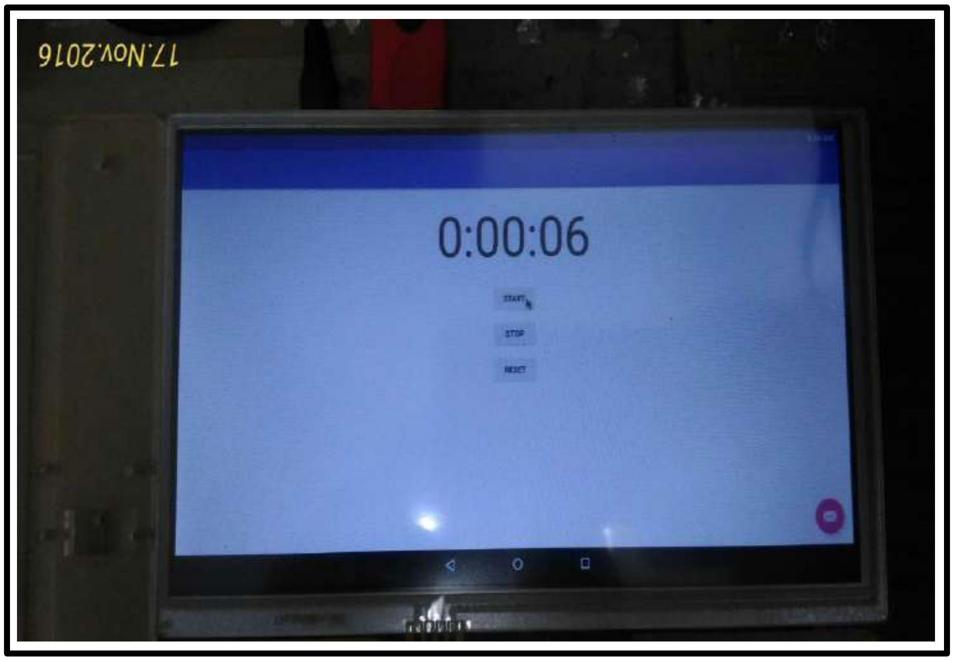


Figure 6.3: Starting the Timer

Stop the timer: The timer stops on clicking the Stop button.

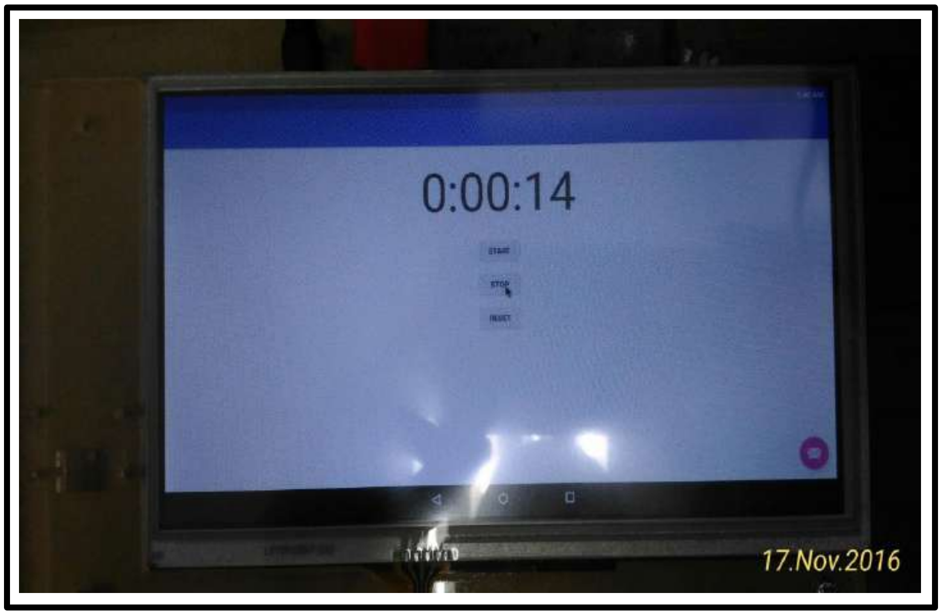


Figure 6.4: Stopping the Timer

Reset the timer: The timer is reset to zero by clicking on Reset button.

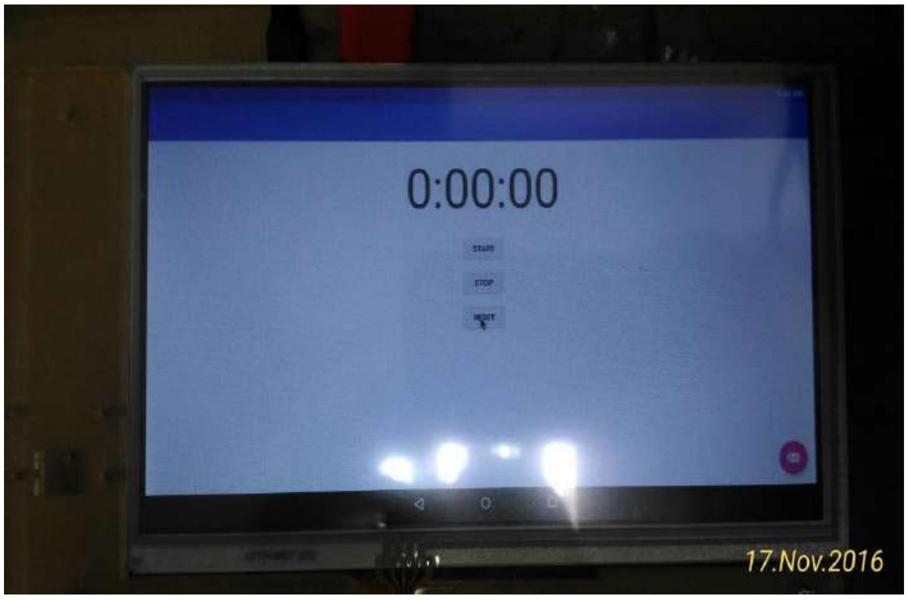


Figure 6.5: Reset the Timer

<u>RESULT:</u> We have successfully deployed the Stop watch app on the Snapdragon Dragon Board 410c.

EXPERIMENT-7

AIM: To design an android app to display all the inbuilt sensors currently supported by dragon board 410c.

COMPONENTS REQUIRED:

Hardware Components: Dragon Board 410c, HDMI to HDMI cable, 2 AC Adapters (12 V), TFT Screen, Driver Board for TFT Screen, Micro USB cable, Mouse.

Software Components: Android Studio.

THEORY:

Dragon Board 410c: The DragonBoard 410c is the first development board based on a Qualcomm Snapdragon 400 series processor. It features advanced processing power, Wi-Fi, Bluetooth connectivity, and GPS, all packed into a board the size of a credit card. Based on the 64-bit capable Snapdragon 410E processor, the Dragon Board 410c is designed to support rapid software development, education and prototyping,

Sensors: A sensor is a device that detects and responds to some type of input from the physical environment. The specific input could be light, heat, motion, moisture, pressure, or any one of a great number of other environmental phenomena. The output is generally a signal that is converted to human-readable display at the sensor location or transmitted electronically over a network for reading or further processing.

Modern mobile phones come with a variety of sensors that automate or easy many of our daily tasks. This field takes into account the presence of an accelerometer, a gyroscope, a compass, and a barometer. Accelerometers in mobile phones are used to detect the orientation of the phone.

Our app will display all the built-in sensors currently supported by the snapdragon 410c along with the version of the sensors and the manufacturing company of the sensor.

PROCEDURE:

1. Create a new project on Android Studio in the Laptop/PC.
2. Name the project as “SENSOR SUPPORT” or any other suitable name.
3. In the MainActivity.java file include the following code:

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <activity
      android:name=".MainActivity"
      android:label="@string/app_name" >
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
```

4. Now we define the activity_main.xml include the following code:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/activity_main"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingLeft="@dimen/activity_horizontal_margin"
```

```
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context="com.example.aaa.sensorssupport.MainActivity">
    <TextView android:text="Sensor " android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textview"
        android:textSize="35dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="DDMD Lab Project"
        android:id="@+id/textView"
        android:layout_below="@+id/textview"
        android:layout_centerHorizontal="true"
        android:textColor="#ff7aff24"
        android:textSize="35dp" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Text"
        android:id="@+id/textView2"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />
</RelativeLayout>
```

5. Now, the application is ready to be run on any android device. To port the application in the snapdragon board, first we have to boot the snapdragon and connect to the TFT screen.
6. Next, connect the snapdragon board to the laptop via a Micro USB cable.
7. On the Android Studio start the compilation of the project. If compilation is successful, the APK file of the project would get transferred to the Snapdragon and the application will get displayed on the TFT screen automatically.

OUTPUT SCREENSHOTS:

Launching the Sensor support app onto Dragon Board but as we can see below there are no built in sensor support in Dragon Board currently because we have not interfaced any sensor with the snapdragon.

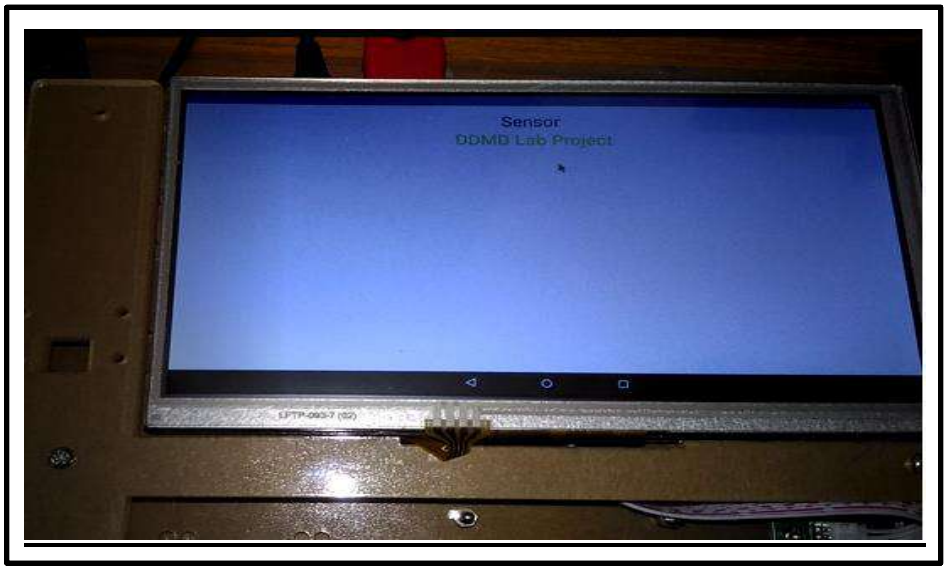
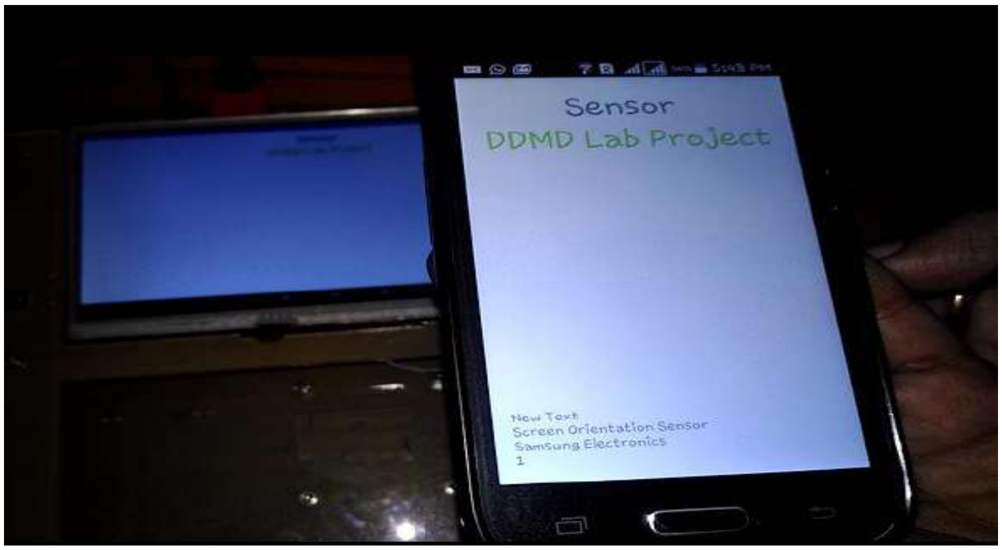


Figure 7.1: Launching Sensor Support app on Dragon Board

Displaying built in sensor of a mobile device using the sensor support app.: The smartphone below has one built in sensor: Screen Orientation Sensor manufactured by Samsung Electronics and is currently has version 1.



New Text
Screen Orientation Sensor
Samsung Electronics
1

Figure 7.2: Displaying built in sensors of a mobile device

<u>RESULT:</u> We have successfully displayed all the inbuilt sensors along with their versions on dragon board 410c.

EXPERIMENT NO-8

AIM: To develop a text-to-speech app using android studio and deploy it on dragon board 410c.

COMPONENTS REQUIRED:

Hardware Components: Dragon Board 410c, HDMI to HDMI cable, 2 AC Adapters (12 V), TFT Screen, Driver Board for TFT Screen, Micro USB cable, Mouse.

Software Components: Android Studio.

THEORY:

Speech synthesis is the artificial production of human speech. A computer system used for this purpose is called a speech computer or speech synthesizer, and can be implemented in software or hardware products. A text-to-speech (TTS) system converts normal language text into speech; other systems render symbolic linguistic representations like phonetic transcriptions into speech.

Android allows you convert your text into voice. Not only you can convert it but it also allows you to speak text in variety of different languages. Android provides Text-To-Speech class for this purpose. In order to use this class, you need to instantiate an object of this class and also specify the initListener.

PROCEDURE:

1. Create a new project on Android Studio in the Laptop/PC.
2. Name the project as “Text-to-speech” or any other suitable name.
3. In the MainActivity.java file include the following code:

```
package com.example.aaa.texttospeech;
import android.app.Activity;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import java.util.Locale;
import android.widget.Toast;
public class MainActivity extends Activity {
   TextToSpeech t1;
   EditText ed1;
   Button b1;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      ed1=(EditText)findViewById(R.id.editText);
      b1=(Button)findViewById(R.id.button);
      t1=new TextToSpeech(getApplicationContext(), new
TextToSpeech.OnInitListener() {
         @Override
         public void onInit(int status) {
            if(status != TextToSpeech.ERROR) {
               t1.setLanguage(Locale.UK);
            }
         }
      });
      b1.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            String toSpeak = ed1.getText().toString();
            Toast.makeText(getApplicationContext(),
toSpeak,Toast.LENGTH_SHORT).show();
            t1.speak(toSpeak, TextToSpeech.QUEUE_FLUSH, null);
            }
        });
    }
    public void onPause(){
        if(t1 !=null){
            t1.stop();
            t1.shutdown();
        }
        super.onPause();
    }
}
```

4. Now we define the activity_main.xml. Type the following code:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context="com.example.aaa.texttospeech.MainActivity">
    <TextView android:text="Text to Speech" android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textview"
        android:textSize="35dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/editText"
        android:layout_marginTop="46dp"
        android:hint="Enter Text"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:textColor="#ff7aff10"
        android:textColorHint="#ffff23d1" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Text to Speech"
        android:id="@+id/button"
        android:layout_below="@+id/editText"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="46dp" />

</RelativeLayout>
```

5. Now, the application is ready to be run on any android device. To port the application in the snapdragon board, first we have to boot the snapdragon and connect to the TFT screen.
6. Next, connect the snapdragon board to the laptop via a Micro USB cable.
7. On the Android Studio start the compilation of the project. If compilation is successful, the APK file of the project would get transferred to the Snapdragon and the application will get displayed on the TFT screen automatically.

OUTPUT SCREENSHOTS:

Launching the Text-to-Speech app icon on the menu:

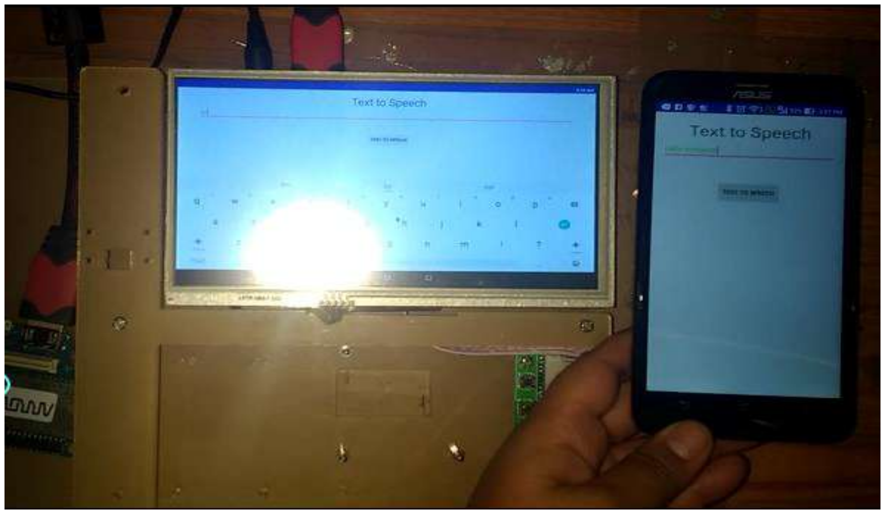


Figure 8.1: Running Text-to-Speech app

RESULT: We have successfully ported the text-to-speech app on dragon board 410c.

REFERENCES

[1]: Design and Development of Smart Phone A Practical Approach by using Raspberry Pi2: Dr. SRN Reddy, Suresh Chande.

[2]: DragonBoard hardware Manual: http://www.seeedstudio.com/document/pdf/DragonBoard_HardwareManual_v5.pdf [Accessed on November 1’2017]

[3]: https://developer.android.com/develop/index.html [Accessed on 5 Nov’17]