

Getting started with Android on the i.MX based COM boards

Embedded Artists AB

Davidshallsgatan 16
SE-211 45 Malmö
Sweden

<http://www.EmbeddedArtists.com>

Copyright 2016 © Embedded Artists AB. All rights reserved.

No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without the prior written permission of Embedded Artists AB.

Disclaimer

Embedded Artists AB makes no representation or warranties with respect to the contents hereof and specifically disclaim any implied warranties or merchantability or fitness for any particular purpose. Information in this publication is subject to change without notice and does not represent a commitment on the part of Embedded Artists AB.

Feedback

We appreciate any feedback you may have for improvements on this document. Send your comments by using the contact form: www.embeddedartists.com/contact.

Trademarks

All brand and product names mentioned herein are trademarks, services marks, registered trademarks, or registered service marks of their respective owners and should be treated as such.

Table of Contents

1	Document Revision History	5
2	Introduction	6
2.1	Conventions.....	6
3	Linux Host Setup	7
3.1	Introduction	7
3.2	Required packages	7
3.3	Configure GIT	7
3.4	Install the JDK	7
3.5	Install the <code>repo</code> tool	7
4	Download sources	9
4.1	Download Android source code	9
4.2	Change GCC version	9
4.3	Download the NXP Linux kernel	9
4.4	Download the u-boot bootloader	10
4.5	NXP patches	10
4.6	Embedded Artists patches	10
4.7	Download Embedded Artists device files	11
5	Building images	13
5.1	Build	13
5.1.1	Build types.....	13
6	Deploying images	14
6.1	Manufacturing tool	14
6.1.1	Download the tool.....	14
6.1.2	Prepare hardware.....	14
6.1.3	OTG boot mode – J2 jumper	14
6.1.4	OTG boot mode – DIP switches	15
6.1.5	Download your own images	16
6.1.6	Run the tool	16
6.1.7	Reset the board.....	17
7	Display options	18
7.1	MCIMX-LVDS1.....	18
7.1.1	LVDS connector	18
8	Power button	20
8.1	iMX6 Quad Developer's Kit.....	20
8.1.1	Example of modification	20
9	Wi-Fi support.....	22

9.1	Connecting to a wireless network	22
9.2	Realtek RTL8188CUS / D-Link DWA-121	24
10	What to do next	25
10.1	Application development	25
11	Android Debug Bridge (adb)	26
11.1	Enable adb support	26
11.2	Connect via network	26
11.3	Take a screenshot	28
11.4	Install an application	28
12	Troubleshooting	29
12.1	Build doesn't finish	29

1 Document Revision History

<i>Revision</i>	<i>Date</i>	<i>Description</i>
A	2016-02-15	First release.
B	2016-05-23	iMX7D uCOM Board / Kit supports Android
C	2016-06-07	Added chapter 9 – Wi-Fi support Updated section 4.6 with new patches needed to add Wi-Fi support.

2 Introduction

This document provides you with step-by-step instructions to download sources, apply patches, and build images needed to create an Android™ system that will run on the i.MX6 SoloX Developer's Kit (**EAK00245**), the i.MX6 Quad Developer's Kit (**EAK00251**) or the i.MX7 Dual uCOM Developer's Kit (**EAK00268**).

The instructions in this document are based on NXP's *Android™ User's Guide* and the Android version used is **5.1.1 (Lollipop)**.

Additional useful documentation:

- *NXP's Android™ User's Guide* – The document can be downloaded from [NXP's website](#).
- The *Getting Started* document for the board you are using.
- [Working with Yocto to build Linux](#) – This document contains instructions of how to setup a virtual machine running Linux.

2.1 Conventions

A number of conventions have been used throughout to help the reader better understand the content of the document.

`Constant width text` – is used for file system paths and command, utility and tool names.

```
$ This field illustrates user input in a terminal running on the  
development workstation, i.e., on the workstation where you edit,  
configure and build Linux
```

```
# This field illustrates user input on the target hardware, i.e.,  
input given to the terminal attached to the COM Board
```

```
This field is used to illustrate example code or excerpt from a  
document.
```

```
This field is used for important notes and highlights
```

3 Linux Host Setup

Note: If you want to program the board with **pre-compiled** Android images and not build the images yourself go to chapter 6 for more information.

3.1 Introduction

Building an Android system requires a Linux host machine. You can either run this host as a standalone / native computer or as a virtual machine on, for example, a Microsoft Windows PC. Please note that a 64-bit environment is needed and it is recommended to use Ubuntu 14.04 since this is the most tested Linux distribution and also the distribution used when writing these instructions.

A list of minimum requirements is available at source.android.com/source/requirements.html. At least 150 GB of free disk space is, for example, needed for a single build.

- Chapter 8 in the document “[Working with Yocto to build Linux](#)” describes how to setup Ubuntu 14.04 as a virtual machine.

3.2 Required packages

Several packages must be installed on the host machine to be able to build the Android system.

```
$ sudo apt-get install git-core gnupg flex bison gperf build-essential zip curl zlib1g-dev gcc-multilib g++-multilib libc6-dev-i386 lib32ncurses5-dev x11proto-core-dev libx11-dev lib32z-dev ccache libgl1-mesa-dev libxml2-utils xsltproc unzip

$ sudo apt-get install uuid uuid-dev zlib1g-dev liblz-dev liblzo2-2 liblzo2-dev lzop git-core curl u-boot-tools mtd-utils android-tools-fsutils
```

3.3 Configure GIT

If GIT hasn't been configured on your computer follow the instructions below. Change “Your name” to your actual name and “Your e-mail” to your e-mail address.

```
$ git config --global user.name "Your name"
$ git config --global user.email "Your e-mail"
```

3.4 Install the JDK

A Java Development Kit (JDK) must be installed to build Android. Java 7 is needed for Lollipop (Android 5) which is the version used in these instructions.

```
$ sudo apt-get install openjdk-7-jdk
```

3.5 Install the repo tool

The `repo` tool has been developed to make it easier to manage multiple Git repositories. Instead of downloading each repository separately the `repo` tool can download all with one instruction. Download and install the tool by following the instructions below.

1. Create a directory for the tool. The example below creates a directory named `bin` in your home folder.

```
$ mkdir ~/bin
```

2. Download the tool

```
$ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
```

3. Make the tool executable

```
$ chmod a+x ~/bin/repo
```

4. Add the directory to the PATH variable as shown below. It is recommended to also add this line to your “.bashrc” file. The path will then be available in each started shell/terminal and also after a reboot of your computer.

```
$ export PATH=~/bin:$PATH
```


4 Download sources

4.1 Download Android source code

These instructions show how to download all the Android source code.

1. Create a directory for the Android sources. In the example below the directory is called "myandroid".

```
$ cd ~  
$ mkdir myandroid  
$ cd myandroid
```

2. Initialize repo. The file containing all needed repositories is downloaded in this step.

```
$ repo init -u https://android.googlesource.com/platform/manifest  
-b android-5.1.1_r1
```

3. Start to download files

```
$ repo sync
```

Note: There are a lot of repositories being downloaded so this step can take several hours.

4.2 Change GCC version

The toolchain used in the Android 5.1.1 release is GCC4.8, but this version has issues compiling the Linux kernel used with the iMX processors. In this section it is described how to change to an earlier version of GCC.

1. Go to the toolchain directory

```
$ cd ~/myandroid/prebuilts/gcc/linux-x86/arm
```

2. Download the arm-eabi-4.6 repository

```
$ git clone  
https://android.googlesource.com/platform/prebuilts/gcc/linux-  
x86/arm/arm-eabi-4.6
```

3. Checkout the branch containing the GCC version to use

```
$ cd arm-eabi-4.6  
$ git checkout android-4.4.3_r1
```

4.3 Download the NXP Linux kernel

Follow these instructions to download and checkout the Linux kernel to use.

Note: The Linux kernel repository is large and can take a long time to download.

```
$ cd ~/myandroid  
$ git clone git://git.freescale.com/imx/linux-2.6-imx.git  
kernel_imx
```

```
$ cd kernel_imx
$ git checkout 15.1.1_2.1.0-ga
```

4.4 Download the u-boot bootloader

Follow these instructions to download and checkout the u-boot version to use.

```
$ cd ~/myandroid/bootable/bootloader
$ git clone git://git.freescale.com/imx/uboot-imx.git uboot-imx
$ cd uboot-imx
$ git checkout 15.1.1_2.1.0-ga
```

4.5 NXP patches

NXP is providing a lot of patches that must be applied to the Android sources in order for the Android system to work on the iMX processors.

1. Create a directory for the patches. The directory created below is only a suggestion. You can change to a different directory.

```
$ cd ~
$ mkdir nxp_patches
$ cd nxp_patches
```

2. Download the patches and put the file into the created directory

https://www.nxp.com/webapp/sps/download/license.jsp?colCode=IMX6_L5.1_2.1.0_AND_SOURCE_BSP&appType=file1&location=null&DOWNLOAD_ID=null

3. Unpack the patches.

```
$ tar -xzvf IMX6_L5.1_2.1.0_AND_SOURCE_BSP.tar.gz
$ cd android_L5.1.1_2.1.0-ga_core_source/code
$ tar -xzvf L5.1.1_2.1.0-ga.tar.gz
```

4. Prepare to apply the patches

```
$ cd ~/myandroid
$ source ~/nxp_patches/android_L5.1.1_2.1.0-ga_core_source/code/L5.1.1_2.1.0-ga/and_patch.sh
```

5. Patch the sources.

```
$ c_patch ~/nxp_patches/android_L5.1.1_2.1.0-ga_core_source/code/L5.1.1_2.1.0-ga/ ea_imx_L5.1.1_2.1.0-ga
```

6. If the patch process was successful you should see a success message that looks like below.

```
*****
Success: Now you can build android code for FSL i.MX platform
*****
```

4.6 Embedded Artists patches

To get Android running on the Embedded Artists i.MX based boards a few more patches must be applied.

1. Create a directory for the patches. The directory created below is only a suggestion. You can change to a different directory.

```
$ cd ~  
$ mkdir ea_patches  
$ cd ea_patches
```

2. Download the Embedded Artists patches and put in the created directory

```
$ wget  
http://imx.embeddedartists.com/common/android_patches_5.1.1.tar.gz
```

3. Unpack the patches

```
$ tar -xzf android_patches_5.1.1.tar.gz
```

4. Apply patches to Linux kernel

```
$ cd ~/myandroid/kernel-imx  
$ patch -p1 < ~/ea_patches/kernel_3.14.52.patch  
$ patch -p1 < ~/ea_patches/remove_gpio_keys.patch
```

5. Apply patch to u-boot bootloader

```
$ cd ~/myandroid/bootable/bootloader/uboot-imx  
$ patch -p1 < ~/ea_patches/uboot_2015.04.patch
```

6. Apply patch to “hardware/imx” directory. Support for the WM8731 codec is added in this step.

```
$ cd ~/myandroid/hardware/imx  
$ patch -p1 < ~/ea_patches/hardware_imx_5.1.1.patch
```

7. Apply patch to “hardware/realtek” directory. Needed to enable Wi-Fi support.

```
$ cd ~/myandroid/hardware/realtek  
$ patch -p1 < ~/ea_patches/hardware_realtek.patch
```

8. Apply patch to “frameworks/opt/net/wifi” directory. Needed to enable Wi-Fi support

```
$ cd ~/myandroid/frameworks/opt/net/wifi  
$ patch -p1 < ~/ea_patches/frameworks_opt_net_wifi.patch
```

4.7 Download Embedded Artists device files

The device files for the Embedded Artists iMX boards are available in a Github repository. The instructions in this section describe how to download these files.

1. Go to the device directory

```
$ cd ~/myandroid/device
```

2. Download files from repository

```
$ git clone  
https://github.com/embeddedartists/ea\_android\_devices.git  
embeddedartists
```

3. Checkout the 5.1.1 branch

```
$ cd embeddedartists  
$ git checkout imx-L5.1.1_2.1.0
```

5 Building images

When the source code has been downloaded and all patches applied it is time to build the necessary images for the Android system.

5.1 Build

Follow the instructions below to build all needed images.

1. Set up the environment

```
$ cd ~/myandroid
$ source build/envsetup.sh
```

2. Select target. The “lunch” command will show a list of all available targets. A portion of the list is shown below. Select the target to build by specifying the number before the target name, such as 24 for “imx6sxea_com_kit-user”. This target means building the build type “user” for the iMX6 SoloX Developer’s Kit. See section 5.1.1 for a description of build types.

```
$ lunch

...
22. aosp_manta-userdebug
23. imx6sxea_com_kit-eng
24. imx6sxea_com_kit-user
25. imx6qea_com_kit-eng
26. imx6qea_com_kit-user
27. sabresd_6dq-eng
...
```

3. Build the images using `make`. “Make” can run several tasks in parallel using the `-jN` argument. If your host computer has several CPUs / cores and/or threads it is recommended to use the `-jN` argument to speed up the build process. In the example below 4 tasks are used.

```
$ make -j4
```

4. When the build has finished the images will be available in the directory below (if the selected target was `imx6sxea_com_kit`).

```
~/myandroid/out/target/product/imx6sxea_com_kit
```

Note: Building the Android related images can take hours depending on the capabilities of your computer.

5.1.1 Build types

Build type	Description
user	Suitable for production. Minimal debug options.
eng	This is a development configuration with debugging options.

6 Deploying images

6.1 Manufacturing tool

NXP's Manufacturing Tool (MFGTool) can be used to write images to the board. This tool is sending files and instructions over USB and the board must be set in OTG boot mode for it to work.

At the moment the tool is only available for Microsoft Windows.

6.1.1 Download the tool

Download the zip file containing the manufacturing tool from <http://imx.embeddedartists.com/>. The name of the file starts with "mfgtools_android_" and contains pre-compiled Android images.

Unpack this zip file somewhere on your computer running Microsoft Windows. Below is a description of some of the content in the zip file.

- mfgtool (root): Contains the actual tool as well as vbs files which can be used to run a specific download configuration.
- mfgtool/Profiles/Linux/OS Firmware/ucl2.xml: This file contains the actual download configurations.
- mfgtool/Profiles/Linux/OS Firmware/files: Contains **pre-compiled** versions of images. The tool will look in this directory when selecting images to download to the board.

6.1.2 Prepare hardware

Begin by reading the *Getting Started* document for the board you are using. This document shows how to setup the board and also gives an overview of the hardware.

The next step is to put the board into OTG boot mode. If you have an early version of the iMX6 SoloX Developer's Kit, that is, a version with a DIP switch mounted as shown in Figure 2, read section 6.1.4 for instructions. Read section 6.1.3 if you have another iMX based developer's kit or a new version of the iMX6 SoloX Developer's kit.

6.1.3 OTG boot mode – J2 jumper

To download images using the manufacturing tool the board must be put into OTG boot mode.

This is accomplished by closing the J2 jumper on the Carrier board; see Figure 1 to locate the jumper. Please note that in the figure the jumper is in open state which means that the COM board will boot from eMMC.

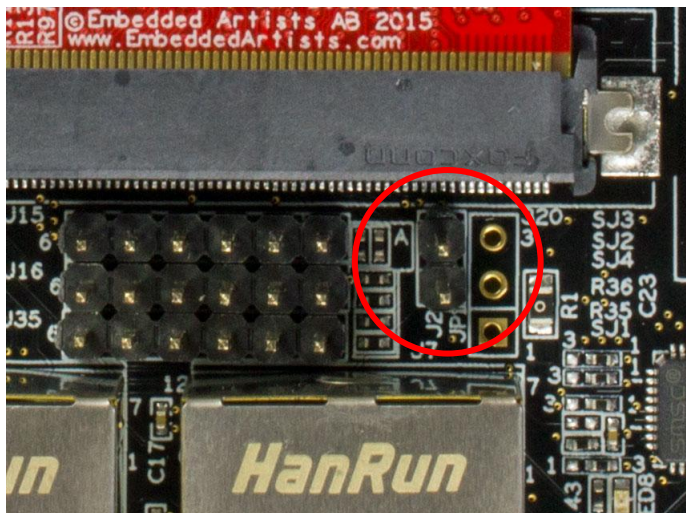


Figure 1 - J2 jumper (opened state)

6.1.4 OTG boot mode – DIP switches

The first version of the iMX6 SoloX COM boards had boot jumpers (DIP switches) mounted on them, see Figure 2. If you have such a COM board you need to set the boot jumpers as described below to force it into OTG boot mode instead of closing jumper J2 as described above.

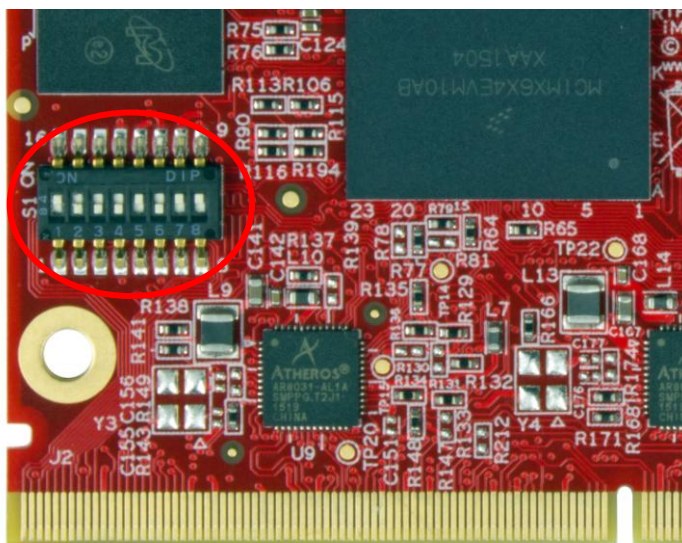


Figure 2 - DIP switch on iMX6 SoloX COM board

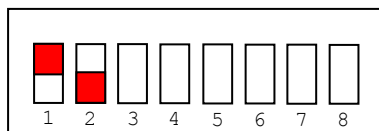


Figure 3 – Boot jumper setting for USB OTG

1. Set the boot jumpers in OTG boot mode as shown in Figure 3. First jumper in the *up* position and the second jumper in *down* position. Please note that the orientation of the DIP switch component can be different on different COM boards. Make sure you are changing the correct jumper by looking at the number by the jumper.
2. Make sure a USB cable is connected between the board (micro-B connector on carrier board) and your PC

3. Reset the board

Note: When you want to boot the software from eMMC you have to reverse the setting, that is, first jumper in *down* position and second jumper in *up* position.

6.1.5 Download your own images

The simplest way to download your own images is to replace the existing file(s) with your file(s), that is, replace the files found in “mfgtool/Profiles/Linux/OS Firmware/files”. If you keep the file names intact the vbs file will download your version of the file.

An alternative is to create a new configuration. In this case you need to open the “uc12.xml” file and then copy an existing configuration (the LIST tag and all its children), give it a new name (change the name attribute) and then modify the instructions so your images are downloaded.

6.1.6 Run the tool

Double click on one of the vbs files to start the manufacturing tool. If the tool can find the board it will write “HID-compliant device” in the status field, see Figure 4 below. If it cannot find the board it will write “No Device Connected”.

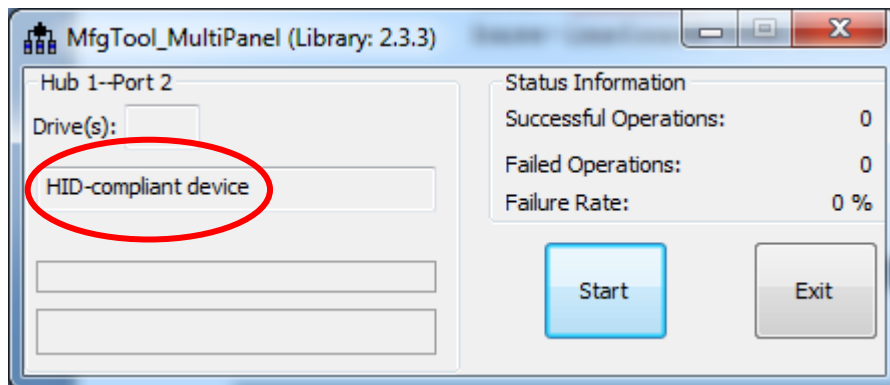


Figure 4 - Manufacturing Tool

Click the Start button to start the download of files. If all operations are successful the progress bars will turn green, see Figure 5. Click the Stop button and then Exit to close the manufacturing tool. If an operation fails the progress bars will turn red. In this case it can be helpful to have a look at the log MfgTool.log which is found in the same directory as the manufacturing tool.

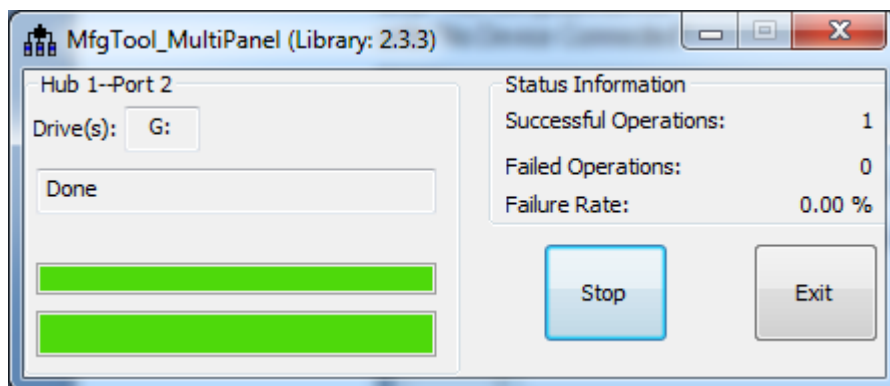


Figure 5 Manufacturing Tool successful download

6.1.7 Reset the board

The images have now been downloaded to the board. The boot configuration must now be changed to booting from eMMC. This is done by removing the J2 boot jumper or reversing the boot jumpers described in section 6.1.4 in case you have the first version of the iMX6 SoloX Developer's Kit.

When the boot configuration has been changed reset the board and the downloaded software will start to execute.

Note: The first time Android is started it can take 5 – 10 minutes before the home screen is shown and you can interact with the device. Subsequent boots will take 30 – 60 seconds depending on the device being used.

7 Display options

7.1 MCIMX-LVDS1

This is a **10 inch** display with a resolution of 1024x768. The display is equipped with a **capacitive touch** screen and is using the **LVDS** interface. This display is used in the default setup of the Android system for Embedded Artists iMX COM boards (as well as for the pre-compiled images available at <http://imx.embeddedartists.com>).

The display can be purchased from, for example, Digi-Key and Mouser.

Digi-Key part number: MCIMX-LVDS1-ND

Mouser part number: 841-MCIMX-LVDS1



Figure 6 – MCIMX-LVDS1 display and Android desktop

7.1.1 LVDS connector

Connect the display to the J23 LVDS connector on the bottom of the COM Carrier board if you are using the iMX6 SoloX Developer's Kit or iMX7 Dual uCOM Developer's Kit. Connect the display to the J24 LVDS connector on the bottom of the COM Carrier board if you are using the iMX6 Quad Developer's Kit. Figure 7 shows where the LVDS connectors are located.

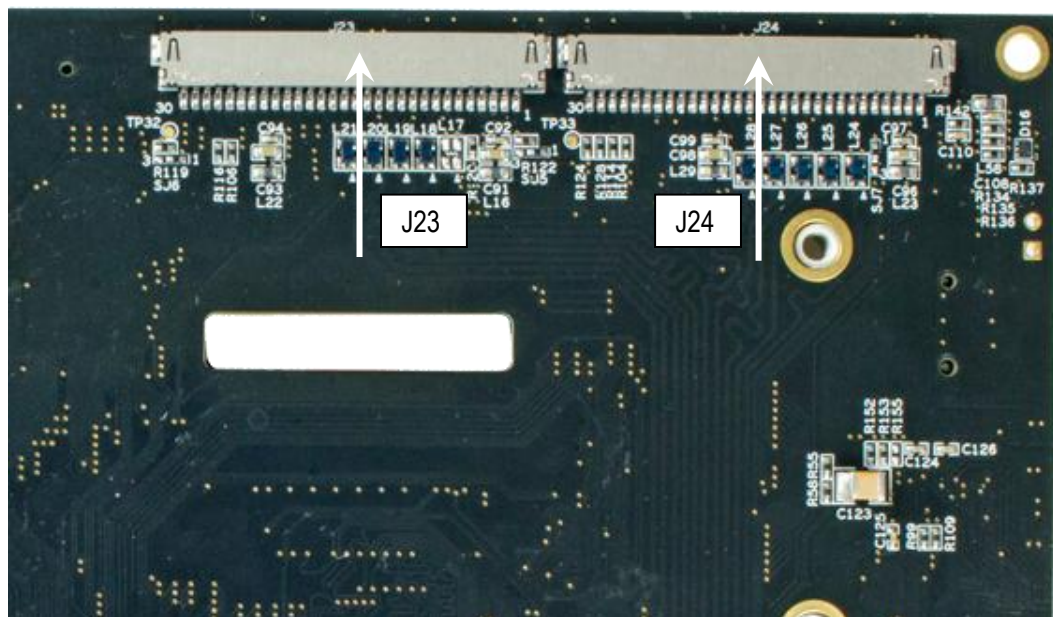


Figure 7 - LVDS connectors

8 Power button

An Android device might need hardware buttons to, for example, turn on/off the device or turn on/off the screen. In the configuration provided by Embedded Artists a “power button” is used to both power on/off the device and “wake” the device, that is, turn on the screen.

On the iMX6 SoloX Developer's Kit and the iMX7 Dual uCOM Developer's Kit the ON/OFF button on the Carrier board is used as power button, see Figure 8.

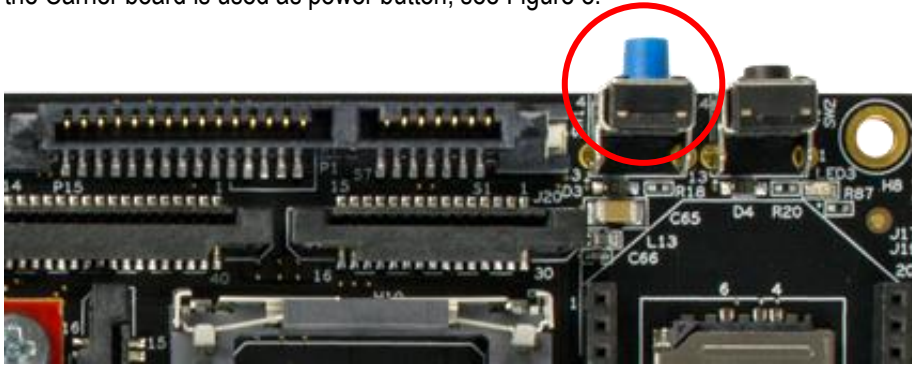


Figure 8 - "Power" button

A quick press on this button turns on the display (if the backlight has been turned off). If the backlight is already on it will be turned off by a quick press on the button. Press the button for a longer time and you will be asked to turn off the device.

8.1 iMX6 Quad Developer's Kit

For the iMX6 Quad Developer's Kit the ON/OFF button unfortunately doesn't function as a power button (from Android's point of view). Instead GPIO3.29 on the iMX6 Quad processor has in the device tree (dts) file been setup as a power key. GPIO3.29 is mapped to GPIO12 on the COM carrier board and this pin is available on expansion connector J29 pin 11, see Figure 9.

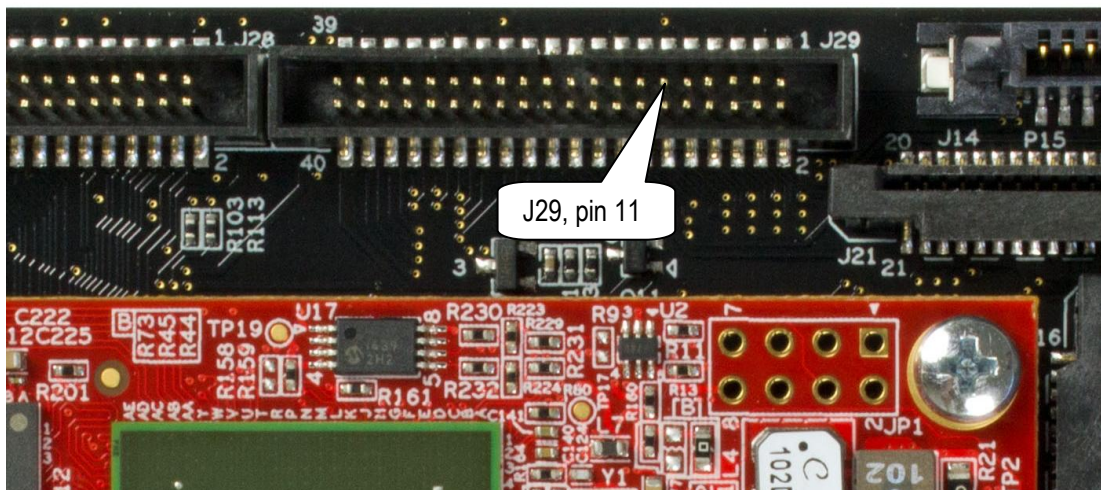


Figure 9 - Pin 11 on J29 expansion connector

8.1.1 Example of modification

With a small and simple rework it is possible to connect the on/off push-button to GPIO3.29. A diode is used to isolate the voltage domains, see schematic below. Any small signal or schottky diode with forward drop voltage below 0.7V can be used.

iMX6 Quad/Dual/DualLite/Solo pin EIM_DATA29-GPIO3_IO29 is connected to COM Carrier board signals GPIO_12.

Figure 10 and Figure 11 below illustrated where to solder the diode and wire to make the connection between the on/off push-button and GPIO3.29.

Push-button for on/off

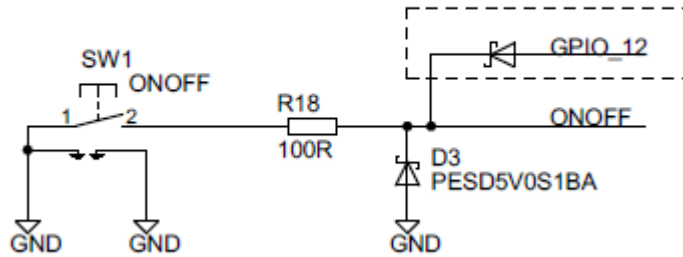


Figure 10 – Schematic showing the suggested rework

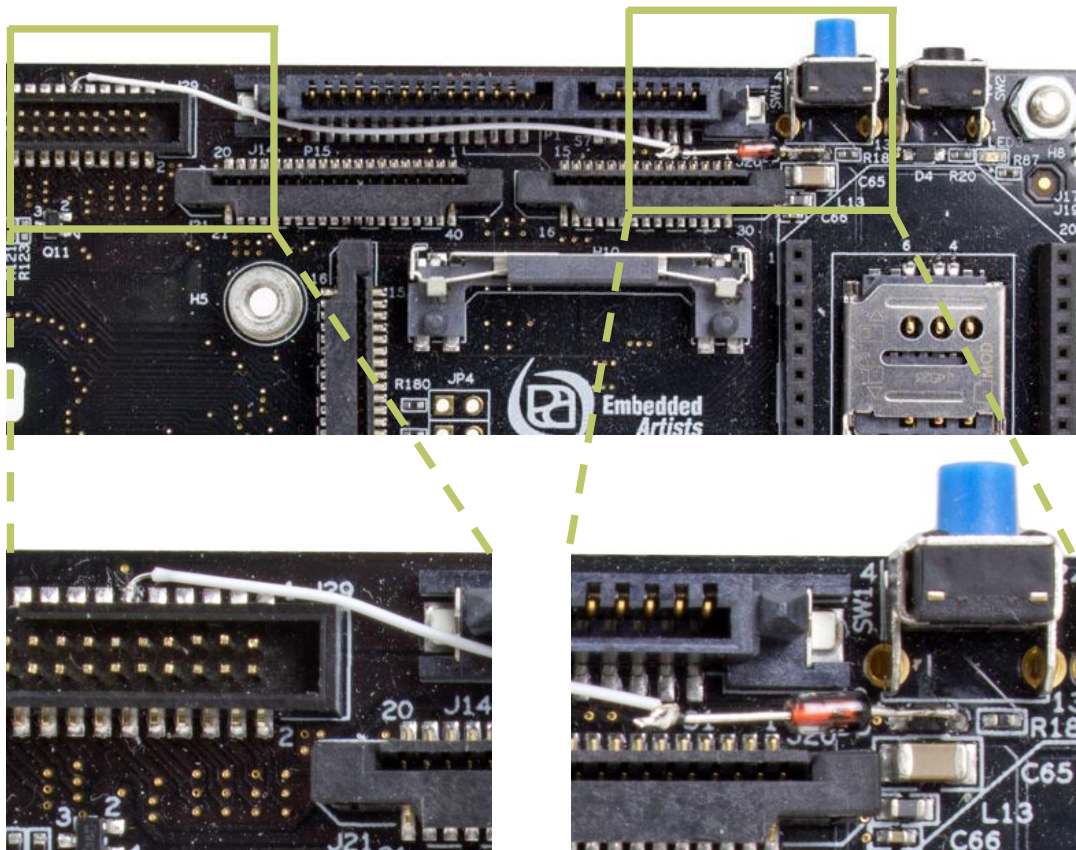


Figure 11 – Photo showing J29, pin 11 connected to the ON/OFF button

9 Wi-Fi support

This chapter describes which modules that can be used to add support for wireless networking to the Android platform.

9.1 Connecting to a wireless network

In Android you can connect to a wireless network from the Settings application as shown in Figure 12.

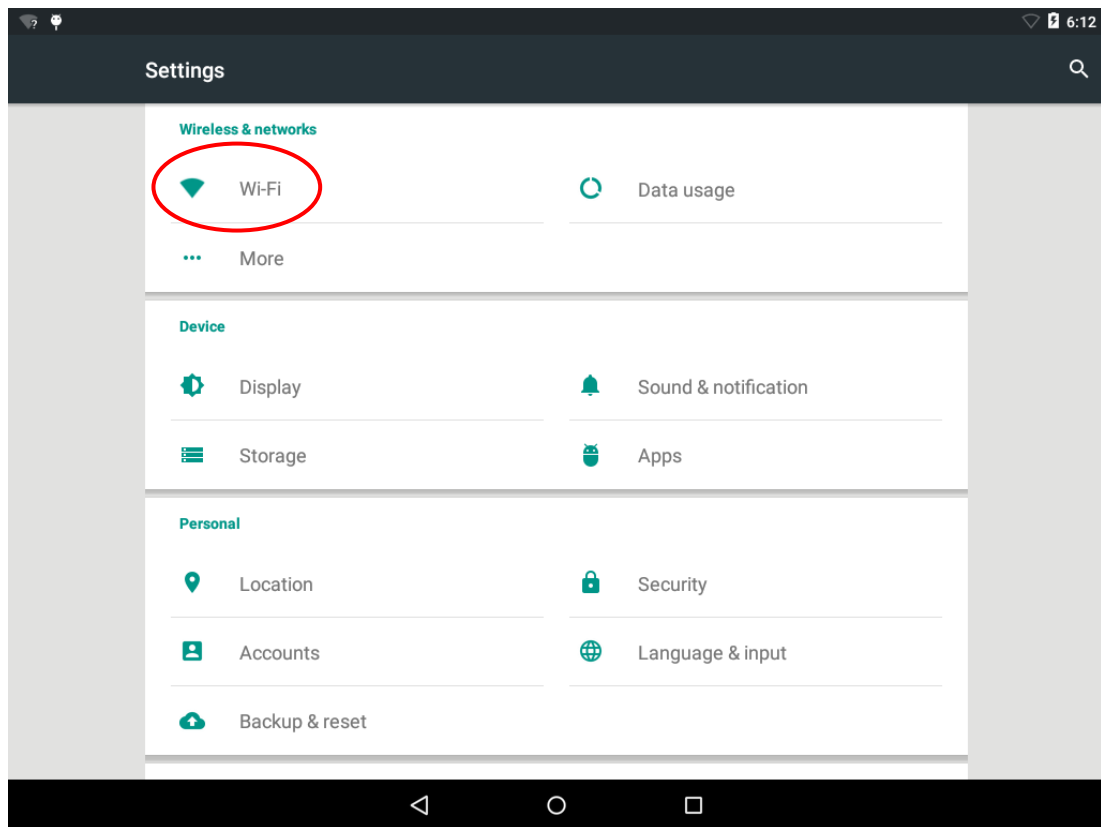


Figure 12 - Settings application

Click on “Wi-Fi” to open the Wi-Fi dialog and then click on the “Off” button to turn on Wi-Fi. When it has been turned on it will scan for wireless networks and list these in the dialog as shown in Figure 13. You can now click on the network that you want to connect to and you will be asked for password as shown in Figure 14.

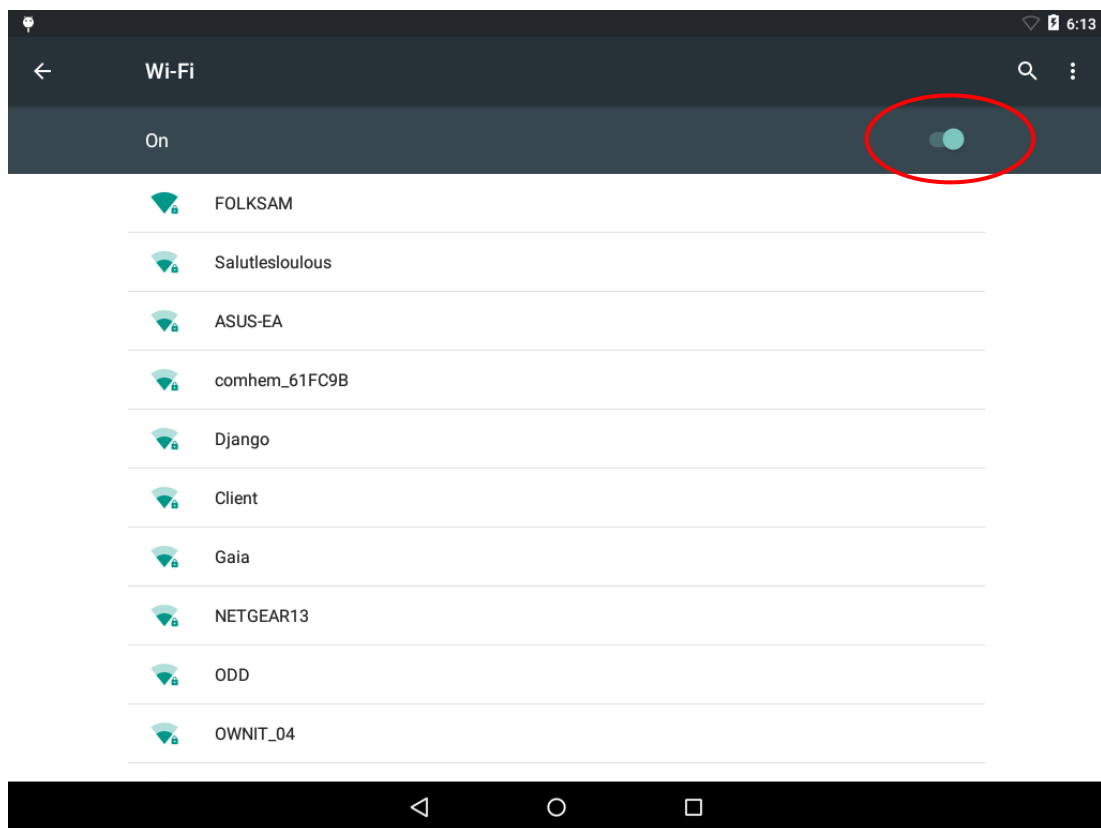


Figure 13 - Wi-Fi settings

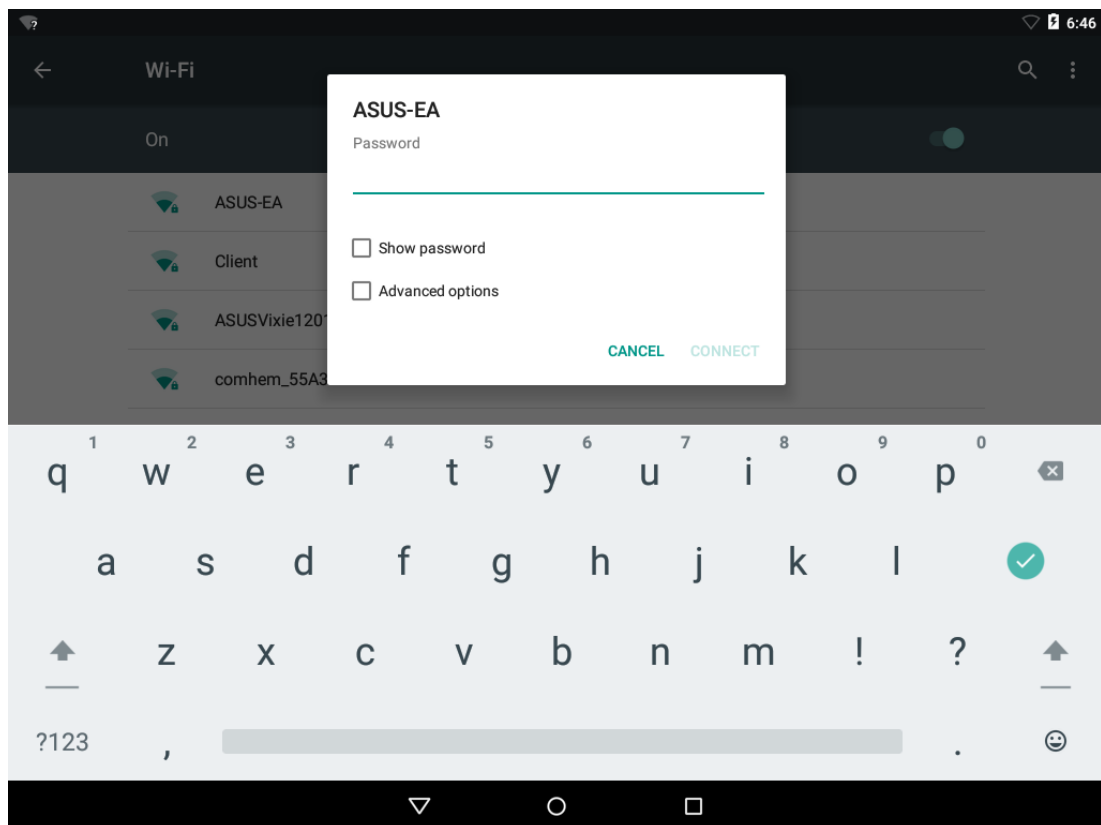


Figure 14 - Wi-Fi password dialog

9.2 Realtek RTL8188CUS / D-Link DWA-121

The pre-built Android images available from Embedded Artists support the Realtek RTL8188CUS Wi-Fi chipset. More specifically the D-Link Wireless N 150 Micro USB adapter (**DWA-121**) has been used as can be seen in Figure 15 below.

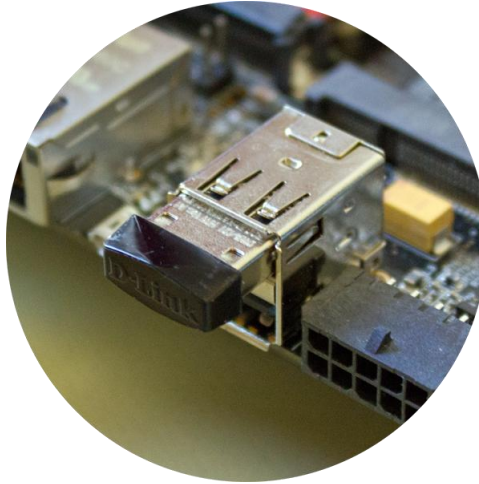


Figure 15 - D-Link DWA-121

10 What to do next

At this point you have an Android system up-and-running on one of our developer's kits. You can now explore the system and play around with the few pre-installed applications.

Please note that *Google Mobile Services*, such as Google Play, Youtube, Gmail, and Google Maps is not open source and must be licensed by Google. These applications are therefore not installed on the system.

Instead of using Google Play you can use the Android Debug Bridge (adb) to install applications, see chapter 11 for more information about adb.

10.1 Application development

It is likely that you want to run your own applications on the system. It is, however, out-of-scope of this document to describe Android application development. A good way to start is to go to [developer.android.com](http://developer.android.com/develop/index.html) which is a site full of useful information for Android application developers.

<http://developer.android.com/develop/index.html>

11 Android Debug Bridge (adb)

The Android Debug Bridge (adb) is a tool that lets you interact with an Android device. It is, for example, possible to retrieve logs or install applications using this tool.

More information about the tool and specifically a list of commands is available at the link below.

<http://developer.android.com/tools/help/adb.html#commandsummary>

11.1 Enable adb support

There is no need for you to do anything to enable debug support via adb since it has been enabled by default in the “init.rc” file for the Embedded Artists boards as can be seen below.

For a final product you want to remove the settings below as well as set the property “ro.adb.secure” to 1.

```
setprop service.adb.tcp.port 5555
setprop persist.service.adb.enable 1
setprop persist.service.debuggable 1
setprop persist.sys.usb.config adb
```

An alternative to enable adb in the “init.rc” file is:

1. Go to “Settings” on the Android device
2. Click “About tablet”
3. Click/touch 7 times on “Build number” and developer options will be enabled
4. Go back to “Settings” and then click on “Developer options”
5. Here you can now enable “USB debugging”

11.2 Connect via network

Using adb via the network is enabled by default. The TCP port is set to 5555 in the “init.rc” file.

1. Make sure you have an Ethernet cable connected to the board. Connect it to the first RJ45 connector as illustrated in Figure 16.
2. Retrieve the IP address, for example, by using the pre-installed “Ethernet App”, see Figure 17 and Figure 18.

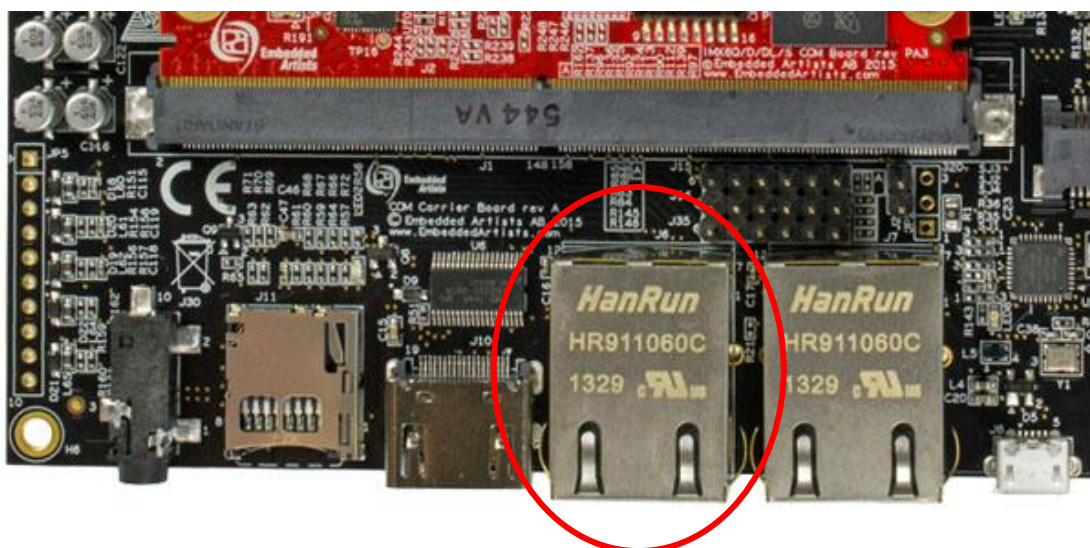


Figure 16 - Ethernet connector on developer's kit

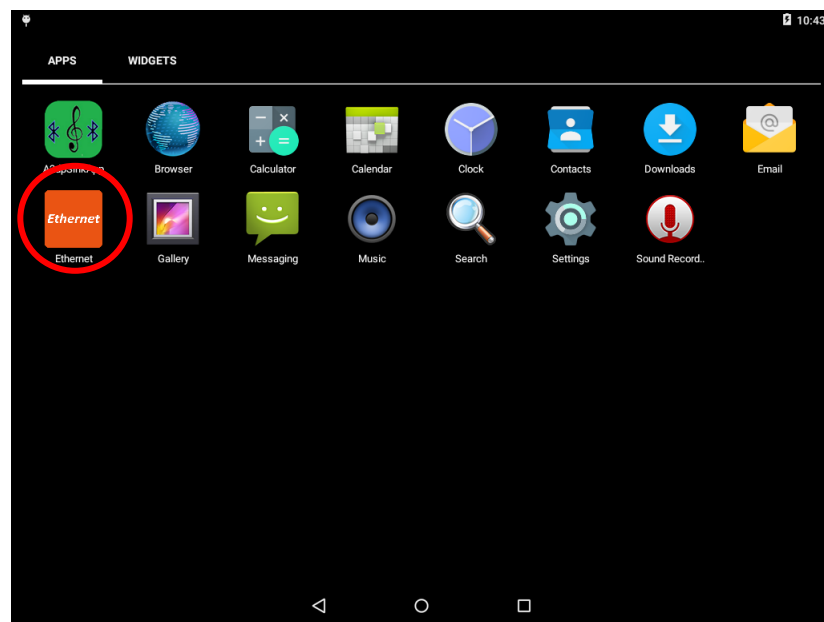


Figure 17 - Android Apps

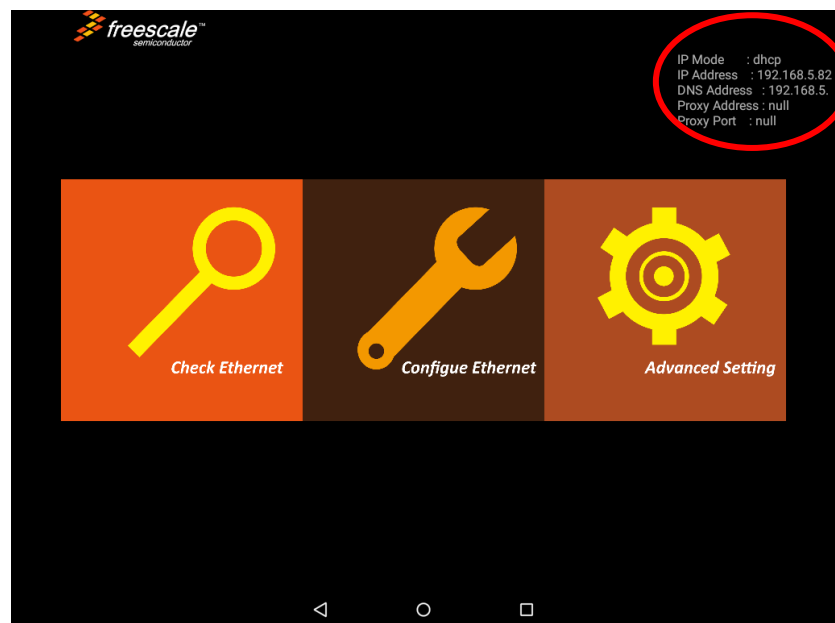


Figure 18 - Ethernet App with IP address

3. Connect to the board

```
$ cd ~/myandroid
$ cd out/host/linux-x86/bin
$ ./adb connect 192.168.5.82
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
connected to 192.168.5.82:5555
```

4. If the connect attempt fails you could try to kill the adb server and then connect again

```
$ ./adb kill-server
```

5. If you have connected successfully you can now continue to use adb to interact with the device.

11.3 Take a screenshot

It is possible to, for example take screenshots or record the device screen as described at the Android developer site. The instructions below show how to take a screenshot and retrieve it from the device.

<http://developer.android.com/tools/help/shell.html#screencap>

1. First make sure you have connected to the device as described in section 11.2 above.
2. Take the screenshot

```
$ ./adb shell screencap /sdcard/screen.png
```

3. Retrieve the screenshot

```
$ ./adb pull /sdcard/screen.png
```

11.4 Install an application

It is possible to install applications using adb.

1. First make sure you have connected to the device as described in section 11.2 above.
2. You must have downloaded the apk file for the application. Here it is assume that it is named myapp.apk and that it is available in the same directory as the adb utility
3. Install the application

```
$ ./adb install myapp.apk
1640 KB/s (17339850 bytes in 10.321s)
  pkg: /data/local/tmp/myapp.apk
Success
```

4. The icon for that application can be found among the other applications on the Android device.

12 Troubleshooting

12.1 Build doesn't finish

Description

The build never finishes although it seem to have compiled all code and generated the file system. The build log can look similar to below. Running “top” on the computer will also show that the “imgdiff” utility is running.

```
Creating filesystem with parameters:
  Size: 374476800
  Block size: 4096
  Blocks per group: 32768
  Inodes per group: 7632
  Inode size: 256
  Journal blocks: 1428
  Label:
  Blocks: 91425
  Block groups: 3
  Reserved block group size: 23
Created filesystem with 1275/22896 inodes and 76543/91425 blocks
build_verity_tree -A
aee087a5be3b982978c923f566a94613496b417f2af592639bc80d141e34dfe7
out/target/product/imx6qea_com_kit/obj/PACKAGING/systemimage_intermediate
s/system.img
/tmp/tmpwCquNy_verity_images/verity.imgsystem/extras/verity/build_verity_
metadata.py 374476800 /tmp/tmpwCquNy_verity_images/verity_metadata.img
4c7475b2519fd4f70a3e75cd504a59ec7fb06cf3d467faede48ccd5b9fe671d9
aee087a5be3b982978c923f566a94613496b417f2af592639bc80d141e34dfe7
/dev/block/mmcblk3p5 out/host/linux-x86/bin/verity_signer
build/target/product/security/verity.pk8
append2simg
out/target/product/imx6qea_com_kit/obj/PACKAGING/systemimage_intermediate
s/system.img /tmp/tmpwCquNy_verity_images/verity_metadata.img
append2simg
out/target/product/imx6qea_com_kit/obj/PACKAGING/systemimage_intermediate
s/system.img /tmp/tmpwCquNy_verity_images/verity.img
```

Reason

The “imgdiff” utility never finishes when creating recovery image. The reason why it never finishes is not known when writing these instructions.

Workaround

Disable “recovery image” creation. Exchange <board directory> below with the name of the board you are using

1. Open “device/embeddedartists/<board directory>/BoardConfig.mk”
2. Set TARGET_NO_RECOVERY to true as shown below

```
BOARD_NOT_HAVE_MODEM := true
BOARD_HAVE_IMX_CAMERA := false
TARGET_NO_RECOVERY := true
```