Oxalis
SDK Developer Manual

Document:

Document:	SDK Developer Manual	Content of the document: Steps to get the source code, customize and build the Oxalis firmware.
Version:	008	
Creator:	ANT/MSB	
Date:	06.12.2018	
Release Status:	Release	Pages: 20

Table of Contents

List of Figures

List of Tables

No table of figures entries found.

Change index:

Rev.:	Date:	Name:	Adaptation:
001	31.10.2018	ANT/MSB	Kick off Document
002	07.11.2018	GEO	First draft version with Oxalis Yocto build
003	12.11.2018	GEO	Second draft version with NFS Boot Option
004	16.11.2018	GEO	Corrected installation filenames in Section 3.2.2 and 3.2.3
005	27.11.2018	AVR	SATA bootup sequence & PCIe Wifi support
006	01.12.2018	MSB	Formalities
007	03.12.2018	AVR	SYSTART image server details update
008	06.12.2018	ATJ	Modification based on testing

1. Scope of this document

After reading this document customer will be able to build a basic image with all interfaces for Oxalis.

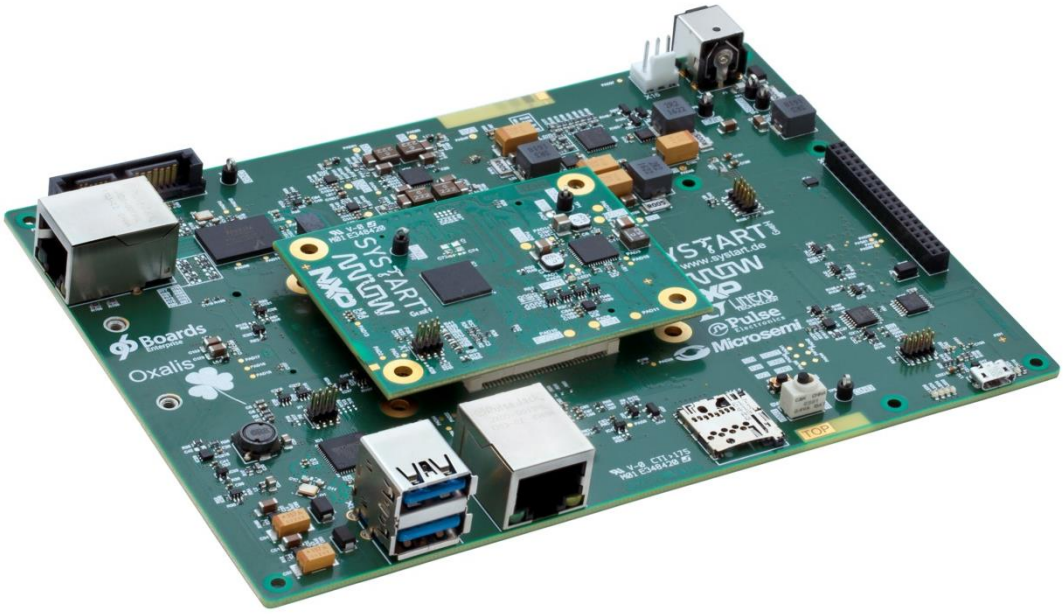

Figure 1: Oxalis with SoM

Oxalis is a modular development board compliant with the 96Boards Enterprise Edition (EE) Platform specification. Developed by SYSTART, the platform consists of a carrier board and a System on Module (SoM) based on the NXP Layerscape LS1012A processor, with a single ARM Cortex-A53 core running up to 800 MHz and 1GB of 16bit DDR3L memory.

Implemented Interfaces:

- 2 Ethernet
- 2 USB 3.0
- 1 PCI-Express
- 1 SD-Card
- 1 SATA
- 4 User LEDs
- Expansion Connector
  - UART
  - SPI
  - I2C
  - GPIO, Reset, Power button

NOTE:

SYSTART provides an expansion board, Oxalis IO breakout board one(BB1) with all the interfaces listed under the expansion connector section above. This expansion connector can be used for small electronic applications and Oxalis interface verifications.

2. Resources and Information

https://www.shop.systart.de/embedded

- Obtaining Oxalis
- Obtaining expansion boards
- Obtaining accessories

https://www.ebs-systart.com/oxalis

- SDK Build Environment and Sources for beginners
- Datasheet
- Binaries
- Links to other resources
- Schematics, Assembly Drawings.

https://github.com/ebs-systart/oxalis

- Developer Information
- Source Codes
- Examples

https://www.96boards.org/products/ee/

- Developer Community and Forum
- Other 96boards Edition compliant resources

https://www.nxp.com/

- SDK Build Environment and Sources

3. Download SDK and Install Build Environment

Oxalis SDK is based on Yocto environment, so it requires a Linux machine to build the Oxalis image. The following instructions assume you are running a Linux machine and are well versed in using a terminal. On any other operating system we recommend installing VirtualBox from

- https://www.virtualbox.org

or any other virtualization system that lets you run a Linux distribution.

NOTE:

Quoting the SDK Documentation: "Yocto Project supports typical Linux distributions: Ubuntu, Fedora, CentOS, Debian, OpenSUSE, etc. More Linux distributions are continually being verified. This SDK has been verified on following Linux distributions: **Ubuntu 14.04, CentOS-7.1.1503, Debian 8.2, Fedora 22 and OpenSUSE 13.2**" You can use other distros, notably a more current LTS Ubuntu, but then you get Warnings to ignore and patches to apply. It will not work out of the box.

There are two variants of SDK

- Free SDK of NXP, which is only available for download with an NXP.com account.
  - Reference SDK of NXP for LS1012a based boards.
  - Supported by NXP
  - SYSTART verified SDK 2.0-1703 version for Oxalis.
- Free SDK of EBS-SYSTART, which is fully based on free SDK of NXP, available for download from the SYSTART server.
  - This is a Snapshot of the SDK 2.0-1703.
  - **SYSTART recommend this SDK** for the very first steps and then switch to the NXP SDK for latest changes.

Major components in the Oxalis Linux Kernel Image are shown in figure below.

Document_Version_V001
Created MSB/23.10.2018; Released STL/23.10.2018

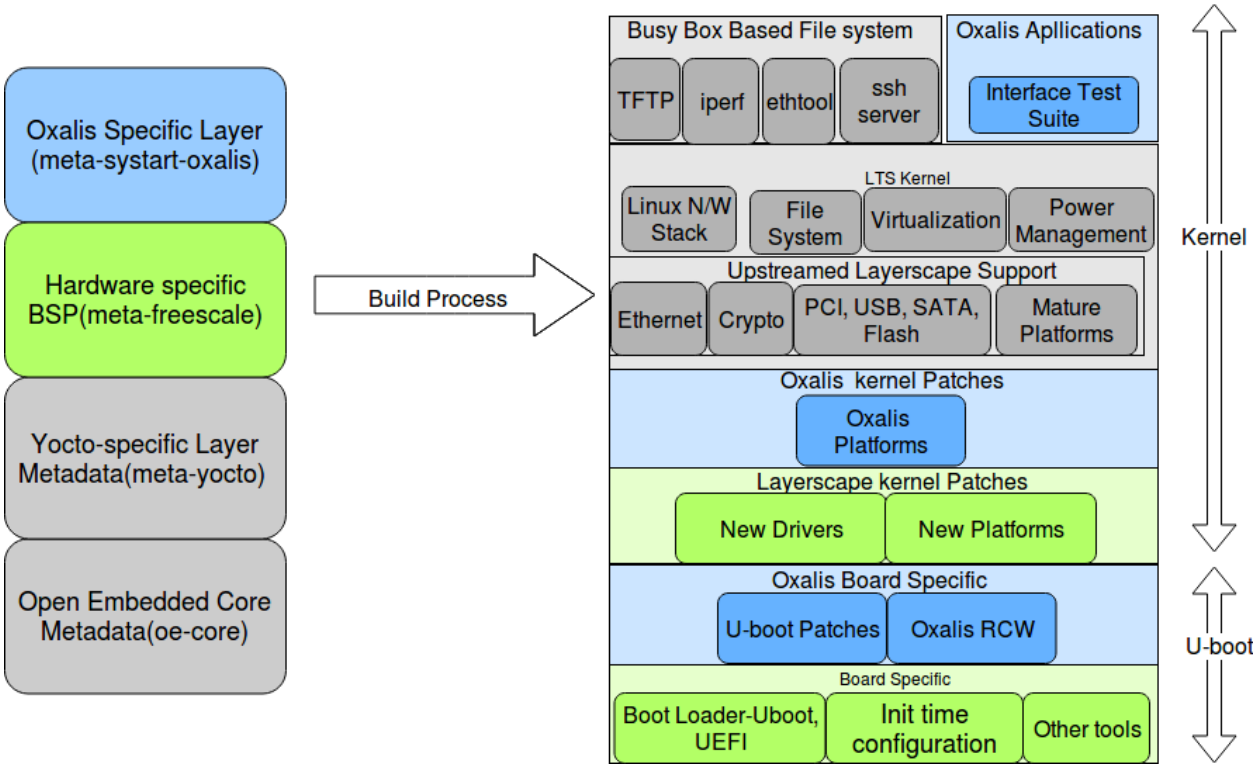

Figure 2: SDK Layers

NOTE

The build steps described in this chapter provides only kernel image. The bootloader (u-boot) is fixed in the current version.

The flow chart below illustrates steps from “download SDK” to “build Linux image” for Oxalis.

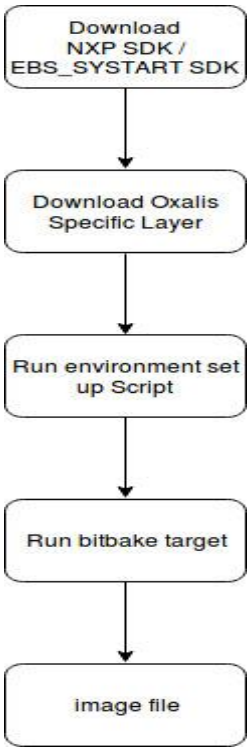

Figure 3: Build Process

Sub sections below describe steps to download and install “NXP SDK” (from the NXP website) and “EBS-SYSTART SDK” from SYSTART server.

This SDK contains:

- Boot loader
- Linux kernel
- User space components
- Tool chain
- Build system
- Package manager.

All of these building blocks are included in the SDK and can be used across many products. The SDK is based upon the standard Linux kernel from http://www.kernel.org, but with patches added by the Yocto Project, NXP, and other community members to support NXP SoCs.

3.1 EBS-SYSTART SDK

The EBS-SYSTART SDK is a Linux-oriented development kit derived from NXP SDK and entire SDK source is present in the SYSTART server.

It allows users to,

1. Evaluate and explore Oxalis board and its features.
2. Develop Linux-based solutions using Oxalis;

3.1.1 Download SDK

- Download SDK from SYSTART website.
    - Download Link

3.1.2 Unzip the SDK

- Unzip the Downloaded SDK

```
$ tar -xzvf oxalis-sdk_stable.tar.gz
```

- Inside the unzipped folder, there will be an ISO image of NXP QorIQ SDK 2.0 & tar file for QorIQ SDK 2.0-1703 update.

3.1.3 Install SDK 2.0

- Mount ISO

```
$  sudo mkdir /mnt/cdrom
```

```
$  sudo mount -o loop 'QorIQ Linux SDK v2.0 SOURCE.iso'
   /mnt/cdrom
```

- Install SDK as a non-root user

```
$  /mnt/cdrom/install
```

Input install path when prompted. SDK will be installed in a directory named **QorIQ-SDK-V2.0-20160527** in the install path.

- Navigate to SDK install directory

```
$  cd QorIQ-SDK-V2.0-20160527
```

- Run host-prepare script to install any dependencies for SDK

```
$  ./sources/meta-freescale/scripts/host-prepare.sh
```

3.1.4 Install SDK 2.0-1703 update

- Extract the tar file

```
$  tar -xjf 'QorIQ Linux SDK v2.0-1703.tar.bz2'
```

When the tar file extraction is completed SDK-V2.0-1703 directory is created.

- Run the install script

```
$  ./SDK-V2.0-1703/install
```

- During the install process, the user will be prompted to input the QorIQ SDK 2.0 ISO installed location,

```
$  <ISO_INSTALL_PATH>/QorIQ-SDK-V2.0-20160527-yocto
```

3.1.5 Add Oxalis Layer to SDK 2.0

- Clone the meta-systart-oxalis layer that provides support for Oxalis.

```
$  cd QorIQ-SDK-V2.0-20160527-yocto/sources/

$  git clone http://gitlab.systart.at-ose.de/systart-dct/meta-
   systart-oxalis.git
```

3.1.6 Build the kernel image for Oxalis

- Run environment setup script for Oxalis

```
$  cd QorIQ-SDK-V2.0-20160527-yocto/sources/meta-systart-
   oxalis/scripts

$  source oxalis-setup-env -m ls1012aoxalis
```

- Accept the EULA, you should now have been transferred into a directory called *build_ls1012aoxalis*
- Run bitbake

```
$  bitbake oxalis-image-kernelitb
```

- A ***.itb*** file (Linux Kernel image) will be created at build_ls1012aoxalis/tmp/deploy/images/ls1012aoxalis/ named as kernel-fsl-ls1012a-oxalis.itb

Use this image file to boot the board using the various ways mentioned in section 4.

3.2 NXP SDK

The NXP SDK is a Linux-oriented development kit. It allows users to

- Evaluate and explore NXP SoC processors' features; experience how they are supported in Linux by working "out of the box" on NXP development boards.
- Develop Linux-based solutions;

Following sections describes the NXP SDK installation sequence with support for building image compatible with Oxalis hardware using NXP SDK 2.0 directly from NXP website.

NOTE:

The steps mentioned below are like the EBS-SYSTART steps except getting the SDK from the NXP website. Users can use below step to build and verify the latest SDK from NXP website. SYSTART verified only using SDK 2.0-1703 version.

3.2.1 Download SDK

- Download iso image for SDK 2.0 from NXP website.
    - Download Link
- Download the tar file for SDK 2.0-1703 update from NXP website

- Download Link

3.2.2 Install QorIQ SDK 2.0

- Mount ISO

```
$  sudo mkdir /mnt/cdrom
```

```
$  sudo mount -o loop QorIQ Linux SDK v2.0 SOURCE.iso /mnt/cdrom
```

- Install SDK as a non-root user

```
$  /mnt/cdrom/install
```

Input install path when prompted.

- Navigate to install directory

```
$  cd <sdk-install-dir>
```

- Run host-prepare script in order to install any dependencies for SDK

```
$  ./sources/meta-freescale/scripts/host-prepare.sh
```

3.2.3 Install SDK 2.0 update

- Extract the tar file

```
$  tar -xjf QorIQ Linux SDK v2.0-1703.tar.bz2
```

- Run the install script

```
$  ./SDK-V2.0-1703/install
```

- During the install process, the user will be prompted to input the SDK 2.0 ISO installed location,

```
$  <ISO_INSTALL_PATH>/QorIQ-SDK-V2.0-20160527-yocto
```

3.2.4 Add Oxalis Layer to Base SDK

- Download the meta-systart-oxalis layer that provides support for Oxalis.

```
$  cd <sdk-install-dir>/sources/
```

```
$  git clone http://gitlab.systart.at-ose.de/systart-dct/meta-
   systart-oxalis.git
```

3.2.5 Build the kernel image for Oxalis

- Run environment setup script for Oxalis

```
$  cd <sdk-install-dir>/sources/meta-systart-oxalis/scripts
```

```
$  source oxalis-setup-env -m ls1012aoxalis
```

- Accept the EULA, you should now have been transferred into a directory called *build_ls1012aoxalis*

- Run bitbake

```
$  bitbake oxalis-image-kernelitb
```

- A ***.itb*** file (Linux Kernel image) will be created at build_ls1012aoxalis/tmp/deploy/images/ls1012aoxalis/ named as kernel-fsl-ls1012a-oxalis.itb

- Use this image file to boot the board using the various ways mentioned in section 0.

4. Deploying images to Oxalis

There are several ways to deploy the image to Oxalis.

- SD Card
- USB Stick
- SATA
- TFTP
- NFS

A detailed description of each method is given in sections 4.1 to 4.5.

4.1 SD-Card

Copy the kernel image created in Section 3.1.6 to a FAT32 formatted SD card and insert the SD card into its slot on Oxalis(X11). Then run the following commands from the u-boot console of Oxalis Board.

```
$  run oxalissettings;run ramargs;

$  setenv bootargs $bootargs init=/sbin/init

$  fatload mmc 0 0x96000000 <filename>.itb

$  pfe stop; bootm 0x96000000
```

4.2 USB

Copy the kernel image created in Section 3.1.6 to a FAT32 formatted USB stick and insert it into one of the USB ports in Oxalis. Then the following commands can be run from the u-boot console of Oxalis.

```
$  run oxalissettings;run ramargs; usb start

$  fatload usb 0 0x96000000 <filename>.itb

$  pfe stop; bootm 0x96000000
```

4.3 SATA

Copy the kernel image created in Section 3.1.6 to a FAT32 formatted hard disk and connect the hard disk to Oxalis's SATA connector(X15) using a SATA (22 Pin) Female to Female Slim-SATA (13 Pin) adapter cable. Then the following commands can be run from the u-boot console of Oxalis.

```
$  run oxalissettings;run ramargs;

$  fatload scsi 0:0 0x96000000 <filename>.itb

$  pfe stop; bootm 0x96000000
```

Oxalis can be booted from a hard disk hat has ext2 filesystem using the following commands.

```
$  run oxalissettings;run ramargs;
```

```
$  ext2load scsi 0:0 0x96000000 <filename>.itb

$  pfe stop; bootm 0x96000000
```

4.4 TFTP

Setup a TFTP server in your host machine and copy the kernel image created in Section 3.1.6 to the *tftpboot* folder of host machine. Connect the host machine to Ethernet port X5(eth0) on Oxalis using a LAN cable. Make sure that both Oxalis and host machine are in same IP domain. Then the following commands can be run from the u-boot console of Oxalis.

```
$  run oxalissettings; run ramargs

$  tftp 0x96000000 <filename>.itb;

$  pfe stop;bootm 0x96000000
```

Oxalis can be connected to the host machine using Ethernet port X3(eth1) also. In that case run the following command before the commands listed above.

```
$  setenv ethact pfe_eth1
```

***Note**: For setting up TFTP server in Linux host machine refer Appendix 1: Setting up TFTP in Host* Machine

4.5 NFS

Setup an NFS server in your host machine and copy the kernel *itb* image created in section 3.1.6 to the *tftpboot* folder of host machine. Connect the host machine to Ethernet port X5(eth0) on Oxalis. Make sure that both Oxalis and host machine are in same IP domain.

Extract the rootfs named **fsl-image-core-ls1012aoxalis.tar.gz** located at **build_ls1012aoxalis/tmp/deploy/images/ls1012aoxalis/** into the NFS export directory on host machine.

Then set following commands can be run from the u-boot console of Oxalis Board.

```
$  setenv bootargs 'console=ttyS0,115200 root=/dev/nfs
   earlycon=uart8250,mmio,0x21c0500 ip=<device ip>:<server
   ip>:<gateway>:<netmask>:<hostname>:<network_interface>:<auto_c
   onfiguration> nfsroot=<serverip>:<path/to/export/directory>'

$  run oxalissettings; tftp 0x96000000 <filename>.itb;

$  pfe stop; bootm 0x96000000:kernel@1 - 0x96000000:fdt@1
```

An example for *bootargs*,

console=ttyS0,115200 root=/dev/nfs earlycon=uart8250,mmio,0x21c0500 ip=192.168.0.230:192.168.0.150:192.168.0.1:255.255.255.0:oxalis:eth0:off nfsroot=192.168.0.150:/tftpboot/oxalis

in this example device ip is 192.168.0.230, server ip 192.168.0.150, gateway *192.168.0.1, network interface eth0 and nfs folder path /tftpboot/oxalis.*

Oxalis can be connected to host machine using ethernet port (X3) also. In that case run following command before the command format listed above

```
$  setenv ethact pfe_eth1
```

Note: *For setting up NFS server in Linux host machine refer* Appendix 2: Setting up NFS Server in Host Machine

5. Customized Kernel

This section describes addition of support for external plug-in devices to Oxalis.

5.1 Mini PCIe Network Card

Oxalis has a mini PCIe M.2 connector which can be used to interface various external devices like Network interface card – both wired and wireless etc. This section gives an example of adding support for Wifi network card (Intel® Centrino® Ultimate-N 6300) to the Oxalis software.

NOTE

The support for Wifi network card (Intel® Centrino® Ultimate-N 6300) is already added in Oxalis Yocto layer. Sections 5.2 to 5.4 is only for reference purpose. Oxalis user with Wifi network card (Intel® Centrino® Ultimate-N 6300) can start following the section 5.4 to start connecting to wireless network.

5.2 Support in Kernel

The default kernel image from NXP does not have support for the Wifi network card by default. The developer needs to add this support by enabling the following configuration in Linux kernel:

```
[*] Networking support --->
    [*] Wireless  --->
      <M>   cfg80211 - wireless configuration API
      [*]    nl80211 testmode command
      [ ]    enable developer warnings
      [ ]    cfg80211 regulatory debugging
      [ ]    enable powersave by default
      [*]    cfg80211 DebugFS entries
      [*]    cfg80211 wireless extensions compatibility
      <M>   Generic IEEE 802.11 Networking Stack (mac80211)
        Default rate control algorithm (Minstrel)  --->
      [ ]   Enable mac80211 mesh networking (pre-802.11s) support
      [ ]   Export mac80211 internals in DebugFS
      [ ]   Trace all mac80211 debug messages
      [ ]   Select mac80211 debugging features  ----
~
```

```
Device Drivers  --->
        [*] Network device support  --->
        --- Network device support
        [*]   Wireless LAN  --->
            --- Wireless LAN
        < >   Marvell 8xxx Libertas WLAN driver support with thin firmware
        < >   Atmel at76c50x chipset  802.11b support
        < >   Atmel at76c503/at76c505/at76c505a USB cards
        < >   Intersil Prism GT/Duette/Indigo PCI/Cardbus (DEPRECATED)
```

```
< >  USB ZD1201 based Wireless device support
< >  Wireless RNDIS USB support
< >  Realtek 8180/8185/8187SE PCI support
< >  Realtek 8187 and 8187B USB support
< >  ADMtek ADM8211 support
< >  Simulated radio testing tool for mac80211
< >  Marvell 88W8xxx PCI/PCIe Wireless support
< >  Atheros Wireless Cards  ----
< >  Broadcom 43xx wireless support (mac80211 stack)
< >  Broadcom 43xx-legacy wireless support (mac80211 stack)
< >  Broadcom IEEE802.11n PCIe SoftMAC WLAN driver
< >  Broadcom IEEE802.11n embedded FullMAC WLAN driver
< >  IEEE 802.11 for Host AP (Prism2/2.5/3 and WEP/TKIP/CCMP)
< >  Intel PRO/Wireless 2100 Network Connection
< >  Intel PRO/Wireless 2200BG and 2915ABG Network Connection
<M>  Intel Wireless WiFi Next Gen AGN - Wireless-N/Advanced-N/Ultimate-N (iwlwifi)
<M>    Intel Wireless WiFi DVM Firmware support
<M>    Intel Wireless WiFi MVM Firmware support
[ ]    Enable broadcast filtering
[ ]    enable U-APSD by default
       Debugging Options  --->
< >  Intel Wireless WiFi 4965AGN (iwl4965)
< >  Intel PRO/Wireless 3945ABG/BG Network Connection (iwl3945)
< >  Marvell 8xxx Libertas WLAN driver support
< >  Hermes chipset 802.11b support (Orinoco/Prism2/Symbol)
< >  Softmac Prism54 support
< >  Ralink driver support  ----
<M>  Realtek rtlwifi family of devices  --->
[ ]  TI Wireless LAN support  ----
< >  ZyDAS ZD1211/ZD1211B USB-wireless support
< >  Marvell WiFi-Ex Driver
< >  CW1200 WLAN support
< >  Redpine Signals Inc 91x WLAN driver support
```

5.3 Support in User space

To connect the Wifi network card to a wireless Access point, the following support needs to be added in user space.

5.3.1 Adding firmware file to rootfs

The Wifi network card requires a firmware file *iwlwifi-6000-4.ucode*. This should be added to the */lib/firmware* in the root filesystem. The file can be obtained from the following location:

https://wireless.wiki.kernel.org/en/users/drivers/iwlwifi

5.3.2 Wireless tools

Wireless tools support shall be added to the root filesystem to manipulate the Wireless Extensions.

- **iwconfig** manipulate the basic wireless parameters
- **iwlist** allow to initiate scanning and list frequencies, bit-rates, encryption keys...
- **iwspy** allow to get per node link quality
- **iwpriv** allow to manipulate the Wireless Extensions specific to a driver (private)
- **ifrename** allow to name interfaces based on various static criteria

The support for wireless tools can be added in Yocto using the following bitbake command.

```
$ bitbake wireless-tools
```

5.3.3 Wpa Supplicant and libnl library

wpa_supplicant is a cross-platform supplicant with support for WEP, WPA and WPA2 (IEEE 802.11i). This is basically needed to authenticate with the wireless access points. The support can be enabled in Yocto with the following bitbake commands

```
$ bitbake wpa-supplicant
```

```
$ bitbake libnl
```

5.4 Connecting to an Access point

The following steps shall be performed to connect the Oxalis board to an external wireless Access Point using the Wifi network card.

5.4.1 Wlan link up

```
$ ifconfig wlan0 up
```

5.4.2 Scan AP list

```
$ iwlist wlan0 scanning
```

5.4.3 Apply authentication settings

```
$ wpa_passphrase <ssid> <passphrase> > /tmp/wpa_supplicant.conf
```

5.4.4 Connect to AP

```
$ wpa_supplicant -B -iwlan0 -c  /tmp/wpa_supplicant.conf  -
Dnl80211
```

5.4.5 Run DHCP client to get IP

```
$ udhcpc -i wlan0
```

6. Appendix 1: Setting up TFTP in Host Machine

Following steps will details the tftp server setting up in host machine running Ubuntu 14.04

Install following packages.

```
$  sudo apt-get install xinetd tftpd tftp
```

Create /etc/xinetd.d/tftp

```
$  sudo nano /etc/xinetd.d/tftp
```

and put this entry

```
service tftp
{
protocol        = udp
port            = 69
socket_type     = dgram
wait            = yes
user            = nobody
server          = /usr/sbin/in.tftpd
server_args     = /tftpboot
disable         = no
}
```

Create a folder /tftpboot this should match whatever you gave in server_args. mostly it will be tftpboot

```
$  sudo mkdir /tftpboot
$  sudo chmod -R 777 /tftpboot
$  sudo chown -R nobody /tftpboot
```

Restart the xinetd service.

```
$  sudo /etc/init.d/xinetd restart
```

You must allow udp port 69 in firewall.

7. Appendix 2: Setting up NFS Server in Host Machine

Following steps will details the NFS server setting up in host machine running Ubuntu 14.04.

Install following package

```
$  sudo apt-get install nfs-kernel-server
```

Open the file /etc/exports

```
$  sudo vim /etc/exports
```

and put the following entry

```
/path/to/export/directory *(rw,no_root_squash,no_subtree_check)
```

Restart the nfs server

```
$  sudo service nfs-kernel-server restart
```