How to configure and build Linux OS with Yocto project for TySOM board

Revision 2.0

April 1st , 2019

Piotr Czak

Zbyszek Tuchewicz

Abstract

Modern embedded systems which based on ARM Cortex processors use also an Operating System. The most popular and free is the Linux OS. This document describes how to build Linux OS for Aldec TySOM boards based on Yocto project.

1 Table of Contents

2 Table of Figures

3 Introduction

The Linux Operating System is very popular in embedded systems and applications which based on the ARM processors, this is why many modern systems like IoT gateways uses the Linux OS. Aldec TySOM boards which based on Xilinx Zynq FPGA with ARM Cortex processor is example of IoT Gateway system. This document describes how to build the Linux OS for Aldec TySOM board and will based on Yocto project which is open source collaboration project. The Yocto Project simplifies way of building custom Linux-based systems for different hardware architectures. More information can be get from *www.yoctoproject.org*.

3.1 Prerequisites

Hardware

- TySOM board:
    - TySOM-3-ZU7EV board
    - TySOM-3A-ZU19EG board
- Hardware configuration for Xilinx Zynq chip with support for HDMI, USB, Ethernet

Software

- Device tree file for hardware configuration
- Internet access
- At least 50Gb free space on hard drive
- Xilinx SDK
- Aldec meta layer
- Reference design from Aldec GitHub version 2018.3
- Appropriate packages installed on the build host. Please refer to *http://www.yoctoproject.org/docs/2.1/mega-manual/mega-manual.html* to get more information about required packages.
- Xilinx Petalinux 2018.3

4 Installation

This chapter describes how to get and install the Yocto project. In this documentation **sumo** branch will be used. Whole installation procedure requires the following steps:

1. Downloading the Yocto project package

First we need to create a new directory called **sumo** for Yocto package and then download package to this directory:

```
git clone –b sumo git://git.yoctoproject.org/poky.git
```

2. Downloading meta-xilinx layer

Now it is time to download meta-xilinx layer into the Yocto project directory. This Xilinx layer contains all necessary configurations for Zynq and Xilinx boards. Go to *sumo/poky* directory and execute:

```
git clone –b rel-2018.3 https://github.com/Xilinx/meta-xilinx.git
```

3. Updating Yocto package with meta-aldec layer

Copy *meta-aldec* folder into *sumo/poky*.

Note that this instructions is based on TySOM-3-ZU7EV SDx platform version 2018.3 but can also be used for TySOM-3A-ZU19EG board.

5 Building Linux

When all packages are downloaded and the Yocto Project directory is updated with **meta-aldec layer** we can start building Linux system. First an environment has to be created. To do so go to *sumo/poky* directory and call:

```
source oe-init-build-env aldec_build
```

This command will create *aldec_build* directory and all necessary files to complete the Linux building process. The following log will be displayed:

```
You had no conf/local.conf file. This configuration file has therefore been
created for you with some default values. You may wish to edit it to, for
example, select a different MACHINE (target hardware). See conf/local.conf
for more information as common configuration options are commented.

You had no conf/bblayers.conf file. This configuration file has therefore been
created for you with some default values. To add additional metadata layers
into your configuration please add entries to conf/bblayers.conf.

The Yocto Project has extensive documentation about OE including a reference
manual which can be found at:
    http://yoctoproject.org/documentation

For more information about OpenEmbedded see their website:
    http://www.openembedded.org/

### Shell environment set up for builds. ###

You can now run 'bitbake <target>'

Common targets are:
    core-image-minimal
    core-image-sato
    meta-toolchain
    meta-ide-support

You can also run generated qemu images with a command like 'runqemu qemux86'
```

Two files have to be updated to use Aldec and Xilinx layers and to build Linux for a specific board:

- aldec _build/conf/bblayers.conf – this file contains list of layers used in building process,
- aldec_build/conf/local.conf – file configures building process itself.

BBLAYERS option in bblayers.conf need to be modified by adding absolute paths to Aldec and Xilinx meta layers.

```
BBLAYERS ?=” \
/space_2/storage/sumo/poky/meta \
/space_2/storage/sumo/poky/meta-poky \
/space_2/storage/sumo/poky/meta-yocto-bsp \
/space_2/storage/sumo/poky/meta-xilinx \
/space_2/storage/sumo/poky/meta-aldec \
“
```

Note that in our case sumo git directory is places in */space_2/storage* directory. In your case it can be different directory.

Three parameters in *local.conf* file should be changed or added:

- **BB_NUMBER_THREADS** – how many threads will be used for bitbake command,
- **PARALLEL_MAKE** – how many threads will be used for source compilation,
- **MACHINE** – for which configuration Linux build will be prepared,
- **EXTRA_IMAGE_FEATURES** – allows extra packages to be added to the generated image.

Exemplary configuration for Aldec TySOM-3-ZU7EV board is presented below:

```
BB_NUMBER_THREADS = "2"
PARALLEL_MAKE = "-j 2"
MACHINE ?= "tysom-3-zu7ev"
EXTRA_IMAGE_FEATURES ?= "tools-debug debug-tweaks"
```

Note: If TySOM-3A-ZU19EG board is used then MACHINE parameter needs to be altered to **tysom-3a-zu19eg**.

Now it is time for Linux building. We will focus in this chapter on two ways: minimal and full command line. To get more information about these version please visit the following pages:

- Minimal version - *http://layers.openembedded.org/layerindex/recipe/579/*
- Full cmdline - *http://layers.openembedded.org/layerindex/recipe/24184/*

1. Building minimal version

This version does not contain any SSH server and is a small image which allows a device to boot. Next two command should be executed in *aldec_build* directory.

```
bitbake core-image-minimal
bitbake device-tree
```

2. Building full cmdline version

This version includes OpenSSH server and includes more full-featured Linux system functionality installed. Next two command should be executed in *aldec_build* directory.

```
bitbake core-image-full-cmdline
bitbake device-tree
```

Note that this process will take approximately 8 hours according to the host workstation and number of used threads.

When building process finished, Yocto builder will automatically create machine directory and storage all necessary files. */space_2/storage/krogoth/poky/aldec_build/tmp/deploy/images/tysom-3-zu7ev/* directory contains files which will be used on SD card. Let’s call this directory ***tysom_build*** directory.

List of the generated files can be similar to the following:

```
rm-trusted-firmware--1.5-xilinx-v2018.3+gitAUTOINC+08560c36ea-r0-20190322161538.bin
arm-trusted-firmware--1.5-xilinx-v2018.3+gitAUTOINC+08560c36ea-r0-20190322161538.elf
arm-trusted-firmware--1.5-xilinx-v2018.3+gitAUTOINC+08560c36ea-r0-20190322161538.ub
arm-trusted-firmware.bin
arm-trusted-firmware.elf
arm-trusted-firmware.ub
atf-uboot.ub
core-image-full-cmdline-tysom-3-zu7ev-20190325111011.rootfs.cpio
core-image-full-cmdline-tysom-3-zu7ev-20190325111011.rootfs.cpio.gz.u-boot
core-image-full-cmdline-tysom-3-zu7ev-20190325111011.rootfs.manifest
core-image-full-cmdline-tysom-3-zu7ev-20190325111011.rootfs.tar.gz
core-image-full-cmdline-tysom-3-zu7ev-20190325111011.testdata.json
core-image-full-cmdline-tysom-3-zu7ev.cpio
core-image-full-cmdline-tysom-3-zu7ev.cpio.gz.u-boot
core-image-full-cmdline-tysom-3-zu7ev.manifest
core-image-full-cmdline-tysom-3-zu7ev.tar.gz
core-image-full-cmdline-tysom-3-zu7ev.testdata.json
core-image-minimal-tysom-3-zu7ev-20190322161538.rootfs.cpio
core-image-minimal-tysom-3-zu7ev-20190322161538.rootfs.cpio.gz.u-boot
core-image-minimal-tysom-3-zu7ev-20190322161538.rootfs.manifest
core-image-minimal-tysom-3-zu7ev-20190322161538.rootfs.tar.gz
core-image-minimal-tysom-3-zu7ev-20190322161538.testdata.json
core-image-minimal-tysom-3-zu7ev.cpio
core-image-minimal-tysom-3-zu7ev.cpio.gz.u-boot
core-image-minimal-tysom-3-zu7ev.manifest
core-image-minimal-tysom-3-zu7ev.tar.gz
core-image-minimal-tysom-3-zu7ev.testdata.json
Image
Image--4.14-xilinx-v2018.3+git0+eeab73d120-r0-tysom-3-zu7ev-20190327124559.bin
Image-tysom-3-zu7ev.bin
modules--4.14-xilinx-v2018.3+git0+eeab73d120-r0-tysom-3-zu7ev-20190327124559.tgz
modules-tysom-3-zu7ev.tgz
tysom-3-zu7ev.dtb
u-boot.bin
u-boot.elf
u-boot-tysom-3-zu7ev.bin
u-boot-tysom-3-zu7ev.elf
u-boot-tysom-3-zu7ev-v2018.01-xilinx-
```

```
v2018.3+gitAUTOINC+d8fc4b3b70-r0.bin
u-boot-tysom-3-zu7ev-v2018.01-xilinx-v2018.3+gitAUTOINC+d8fc4b3b70-r0.elf
uEnv.txt
uEnv-tysom-3-zu7ev.txt
uEnv-tysom-3-zu7ev-v2018.01-xilinx-v2018.3+gitAUTOINC+d8fc4b3b70-r0.txt
```

6 How to expand Linux OS with other layers

Previous chapter describes how to build minimal versions of Linux OS, what sometimes is not enough in IoT applications which have different requirements. This chapter will guide you how to expand this minimal version and add other layers to customize Linux OS.

First we need to download and set new meta layers in *bblayers.conf* file. All layers are listed on webpage: *http://layers.openembedded.org/layerindex/branch/master/layers/*

The Yocto project provides also very comprehensive instructions how to customize Linux OS. This documentation is available on the website: *http://www.yoctoproject.org/docs/1.1/poky-ref-manual/poky-ref-manual.html#usingpoky-extend-customimage*

6.1 Exemplary extended configuration

First we need to download additional meta layers used by new Linux version:

```
git clone git://github.com/OSSystems/meta-browser.git
git clone -b sumo git://git.openembedded.org/meta-openembedded
git clone https://github.com/meta-rust/meta-rust.git
```

Next step is adding new layers into aldec _build/conf/bblayers.conf file.

```
BBLAYERS ?=” \
/space_2/storage/sumo/poky/meta \
/space_2/storage/sumo/poky/meta-poky \
/space_2/storage/sumo/poky/meta-yocto-bsp \
/space_2/storage/sumo/poky/meta-xilinx/meta-xilinx-bsp \
/space_2/storage/sumo/poky/meta-aldec \
/space_2/storage/sumo/poky/meta-rust \
/space_2/storage/sumo/poky/meta-browser \
/space_2/storage/sumo/poky/meta-openembedded/meta-oe \
/space_2/storage/sumo/poky/meta-openembedded/meta-multimedia \
/space_2/storage/sumo/poky/meta-openembedded/meta-networking \
/space_2/storage/sumo/poky/meta-openembedded/meta-gnome \
/space_2/storage/sumo/poky/meta-openembedded/meta-xfce \
/space_2/storage/sumo/poky/meta-openembedded/meta-python \
“
```

The aldec_build/conf/local.conf configuration file need to be modified.

Exemplary configuration for Aldec TySOM-3-ZU7EV board is presented below:

```
BB_NUMBER_THREADS = "2"
PARALLEL_MAKE = "-j 2"
MACHINE ?= "tysom-3-zu7ev"
EXTRA_IMAGE_FEATURES ?= "tools-debug debug-tweaks"

# meta-xfce, avoid dependencies on meta-multimedia:
```

```
BBMASK = "meta-xfce/recipes-multimedia"
# local  mirrors from petalinux
# Add Pre-mirrors of tool
PREMIRRORS_prepend = "git://.*/.* file:///path/to/petalinux-
2018.3/components/yocto/downloads/ \n \
ftp://.*/.* file:///path/to/petalinux-2018.3/components/yocto/downloads/ \n \
http://.*/.* file:///path/to/petalinux-2018.3/components/yocto/downloads/ \n \
https://.*/.* file:///path/to/petalinux-2018.3/components/yocto/downloads/ \n"

#Add Pre-mirrors from petalinux-config
PREMIRRORS_append = " git://.*/.* http://petalinux.xilinx.com/sswreleases/rel-
v2018.3/downloads \n \
ftp://.*/.* http://petalinux.xilinx.com/sswreleases/rel-v2018.3/downloads \n \
http://.*/.* http://petalinux.xilinx.com/sswreleases/rel-v2018.3/downloads \n \
https://.*/.* http://petalinux.xilinx.com/sswreleases/rel-v2018.3/downloads \n"

# enable x11 , disable wayland
DISTRO_FEATURES_append = " x11"
DISTRO_FEATURES_remove = "wayland"
```

Note: If TySOM-3A-ZU19EG board is used then MACHINE parameter needs to be altered to **tysom-3a-zu19eg**.

Execute next two command to build new custom Linux version:

```
bitbake core-image-minimal-xfce
bitbake device-tree
```

7 Preparing SD card

This chapter describes how to prepare micro SD card to use it as boot medium for running embedded Linux on TySOM-3-ZU7EV board. Preferable model of SD card is 16 GB class 10 card. It provides enough disk space and read/write access performance needed to run Desktop.

Important note: always use umount or similar command before removing memory card from the card reader device. Ignoring this step may result in filesystem corruption and you can turn your memory card into a brick.

7.1 Preparing boot and root partitions.

1. Plug in your micro SD card via adapter into card reader or similar device connected to the host workstation;
2. Detect device name assigned to the SD card device by host OS:

```
dmesg | tail -n 10
```

3. Check output for the name similar to sdd:

```
[1996068.063860] sd 10:0:0:2: [sdd] 31116288 512-byte logical blocks: (15.9
GB/14.8 GiB)
[1996068.076930] sdd: sdd1
```

4. Let's assume that assigned name is sdd and full device path is /dev/sdd;

5. Switch to the superuser mode;
6. Erase old partition table:

```
dd if=/dev/zero of=/dev/sdd bs=1024 count=1
```

7. Check current memory card formatting:

```
fdisk -l /dev/sdd
```

8. Run *fdisk* to start making changes on memory card:

```
fdisk /dev/sdd
```

9. The output should be similar to this:

```
Welcome to fdisk (util-linux 2.24.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.
Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x2536c232.
```

10. Make sure there is no partition table. If it exists wipe every partition applying “d” command:

```
Command (m for help): d
Partition number (1-4): 1
```

11. Configure heads, sectors and cylinders, always use default value for the number of cylinders:

```
Command (m for help): x
Expert command (m for help): h
Number of heads (1-256, default 64): 255
Expert command (m for help): s
Number of sectors (1-63, default 32): 63
Expert command (m for help): c
Number of cylinders (1-1048576, default 1936): 1936
Expert command (m for help): r
```

12. Create two primary partitions (if value is not specified default is used):

```
Command (m for help): n
Partition type:
  p primary (0 primary, 0 extended, 4 free)
  e extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-31116287, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-31116287, default 31116287): +50M
Created a new partition 1 of type 'Linux' and of size 50 MiB.

Command (m for help): n
Partition type:
  p primary (1 primary, 0 extended, 3 free)
  e extended
```

```
Select (default p): p
Partition number (2-4, default 2): 2
First sector (104448-31116287, default 104448):
Last sector, +sectors or +size{K,M,G,T,P} (104448-31116287, default 31116287):
Created a new partition 2 of type 'Linux' and of size 14.8 GiB.
```

13. Set partition IDs and bootable flag:

```
Command (m for help): a
Partition number (1,2, default 2): 1
The bootable flag on partition 1 is enabled now.
Command (m for help): t
Partition number (1,2, default 2): 1
Hex code (type L to list all codes): c
Changed type of partition 1 to 'W95 FAT32 (LBA)'.
Command (m for help): t
Partition number (1,2, default 2): 2
Hex code (type L to list all codes): 83
Changed type of partition 2 to 'Linux'.
```

14. Check new partition table and apply changes:

```
Command (m for help): p
Disk /dev/sdd: 14.9 GiB, 15931539456 bytes, 31116288 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x2536c232
Device Boot Start End Blocks Id System
/dev/sdd1 * 2048 104447 51200 c W95 FAT32 (LBA)
/dev/sdd2 104448 31116287 15505920 83 Linux
Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
```

15. Create file systems for both memory card partitions:

```
mkfs.vfat -F 32 -n boot /dev/sdd1
mkfs.ext4 -L root /dev/sdd2
```

16. Now we can mount memory card partition filesystems and operate with them. Let's assume that we have already created directories at */mnt/boot* and */mnt/rootfs*:

```
mount /dev/sdd1 /mnt/boot/
mount /dev/sdd2 /mnt/rootfs/
```

Note that some Linux distros can automatically mount memory card partitions when card is plugged in. In that case there is no need to use mount command, but anyway do not forget to unmount partitions using **umount** command or any other way provided by Linux OS GUI.

7.2 Preparing SD card files

This chapter describes step by step which files should be copied to SD card partitions.

1. Boot partition

First we need to copy files generated by Yocto builder to boot partition:

```
cp <tysom_build>/Image--4.14-xilinx-v2018.3+git0+eeab73d120-r0-tysom-3-zu7ev-
20190327124559.bin /mnt/boot/Image
cp <tysom_build>/uEnv-tysom-3-zu7ev-v2018.01-xilinx-
v2018.3+gitAUTOINC+d8fc4b3b70-r0.txt /mnt/boot/uEnv.txt
cp <tysom_build>/tysom-3-zu7ev.dtb /mnt/boot/uEnv.txt
```

Bootloader image is easy to build using files from reference design:

- copy file *bootbin/yocto_linux.bif* to *TySOM/Sdx-platforms/2018.3/TySOM_3_ZU7/sv* folder,
- set path to tool: source */path/to/Xilinx/SDK/2018.3/settings64.sh*,
- build bootloader:

```
bootgen -w on -arch zynqmp -image yocto_linux.bif -o i BOOT.bin
```

2. Load to RAM

```
cp <tysom_build>/core-image-minimal-tysom-3-zu7ev-
20190322161538.rootfs.cpio.gz.u-boot /mnt/boot/uramdisk.image.gz
```

This file is created in <tysom_build> folder.

3. Root partition

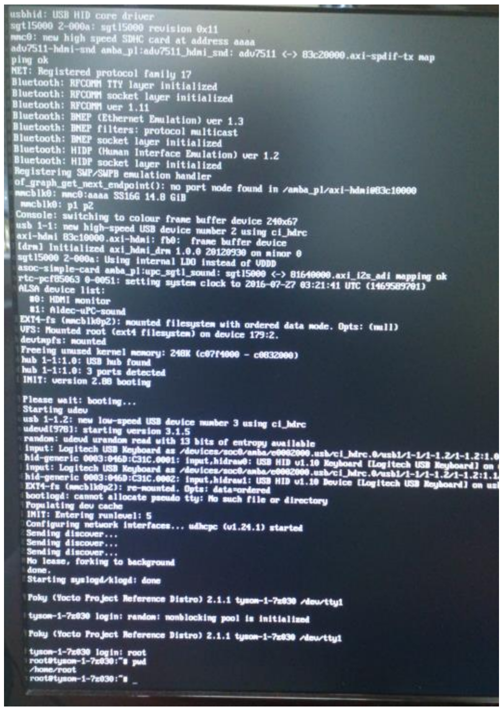
Unpack to /mnt/rootfs files generated by Yocto builder:

```
tar -xf <tysom_build>/core-image-minimal-tysom-3-zu7ev-
20190322161538.rootfs.tar.gz -C /mnt/rootfs
```

Safely unmount filesystems:

```
umount /mnt/boot
umount /mnt/rootfs
```

The SD card is ready to insert to socket and boot The Linux OS. After power off, minimal Linux version will boot in a few seconds and allows used to log in as a root what is presented in Figure 1.



```
usbhid: USB HID core driver
sgtl5000 2-000a: sgtl5000 revision 0x11
mmc0: new high speed SDHC card at address aaaa
adv7511-hdmi-snd amba_pl:adv7511_hdmi_snd: adv7511 <-> 83c20000.axi-spdif-tx map
ping ok
NET: Registered protocol family 17
Bluetooth: RFCOMM TTY layer initialized
Bluetooth: RFCOMM socket layer initialized
Bluetooth: RFCOMM ver 1.11
Bluetooth: BNEP (Ethernet Emulation) ver 1.3
Bluetooth: BNEP filters: protocol multicast
Bluetooth: BNEP socket layer initialized
Bluetooth: HIDP (Human Interface Emulation) ver 1.2
Bluetooth: HIDP socket layer initialized
Registering SWP/SWPB emulation handler
of_graph_get_next_endpoint(): no port node found in /amba_pl/axi-hdmi@83c10000
mmcblk0: mmc0:aaaa SS16G 14.8 GiB
 mmcblk0: p1 p2
Console: switching to colour frame buffer device 240x67
usb 1-1: new high-speed USB device number 2 using ci_hdrc
axi-hdmi 83c10000.axi-hdmi: fb0:  frame buffer device
[drm] Initialized axi_hdmi_drm 1.0.0 20120930 on minor 0
sgtl5000 2-000a: Using internal LDO instead of VDDD
asoc-simple-card amba_pl:upc_sgtl_sound: sgtl5000 <-> 81640000.axi_i2s_adi mapping ok
rtc-pcf85063 0-0051: setting system clock to 2016-07-27 03:21:41 UTC (1469589701)
ALSA device list:
  #0: HDMI monitor
  #1: Aldec-uPC-sound
EXT4-fs (mmcblk0p2): mounted filesystem with ordered data mode. Opts: (null)
VFS: Mounted root (ext4 filesystem) on device 179:2.
devtmpfs: mounted
Freeing unused kernel memory: 248K (c07f4000 - c0832000)
hub 1-1:1.0: USB hub found
hub 1-1:1.0: 3 ports detected
INIT: version 2.88 booting

Please wait: booting...
Starting udev
usb 1-1.2: new low-speed USB device number 3 using ci_hdrc
udevd[978]: starting version 3.1.5
random: udevd urandom read with 13 bits of entropy available
input: Logitech USB Keyboard as /devices/soc0/amba/e0002000.usb/ci_hdrc.0/usb1/1-1/1-1.2/1-1.2:1.0
hid-generic 0003:046D:C31C.0001: input,hidraw0: USB HID v1.10 Keyboard [Logitech USB Keyboard] on
input: Logitech USB Keyboard as /devices/soc0/amba/e0002000.usb/ci_hdrc.0/usb1/1-1/1-1.2/1-1.2:1.1
hid-generic 0003:046D:C31C.0002: input,hidraw1: USB HID v1.10 Device [Logitech USB Keyboard] on us
EXT4-fs (mmcblk0p2): re-mounted. Opts: data=ordered
bootlogd: cannot allocate pseudo tty: No such file or directory
Populating dev cache
INIT: Entering runlevel: 5
Configuring network interfaces... udhcpc (v1.24.1) started
Sending discover...
Sending discover...
Sending discover...
No lease, forking to background
done.
Starting syslogd/klogd: done

Poky (Yocto Project Reference Distro) 2.1.1 tysom-1-7z030 /dev/tty1

tysom-1-7z030 login: random: nonblocking pool is initialized

Poky (Yocto Project Reference Distro) 2.1.1 tysom-1-7z030 /dev/tty1

tysom-1-7z030 login: root
root@tysom-1-7z030:~# pwd
/home/root
root@tysom-1-7z030:~# _
```

Figure 1. Minimal Linux booting.

If you will use the extended custom Linux configuration from 6.1 chapter you will see similar screen to the presented in Figure 2.

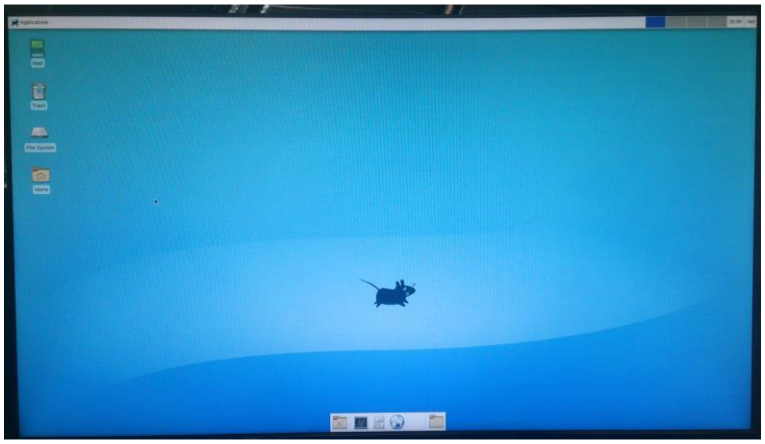


Figure 2. Yocto Linux with UI.

8 About Aldec, Inc.

Established in 1984, Aldec Inc. is an industry leader in Electronic Design Verification and offers a patented technology suite including: RTL Design, RTL Simulators, Hardware-Assisted Verification, Design Rule Checking, IP Cores, DO-254 Functional Verification and Military/Aerospace solutions. Continuous innovation, superior product quality and total commitment to customer service comprise the foundation of Aldec's corporate mission. For more information, visit www.aldec.com.