Introduction to NXP Yocto

October 2020

SoMLabs

SOM
System on Module

CB
Carrier Board

DK
Development Kit

Engineering
Since 2003 delivering proven designs

Agenda

- Introduction to the development virtual machine
- Explaining Yocto project structure
- SoMLabs meta-layer and hardware support
- Building system and SDK
- System installation
- Building C application with Yocto SDK
- Creating a new recipe

Exercises

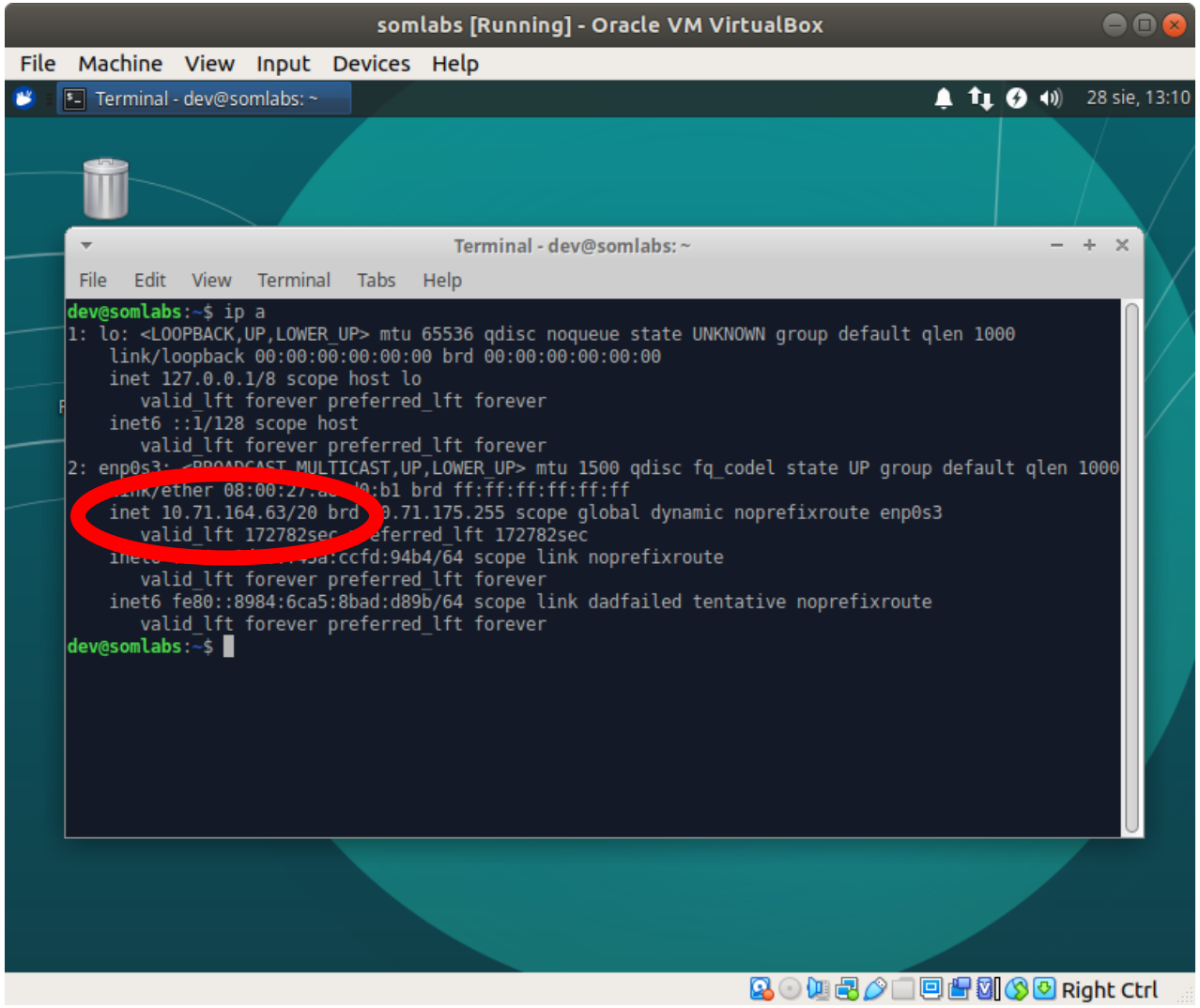
- /home/dev/Excercises/Workshop1/
- Lab1 - Building C application with SDK
- Lab2 - Adding a new recipe
- Lab3 - Run application on system boot

SoMLabs virtual machine

SoMLabs virtual machine

- Oracle VirtualBox 6.1 with Extension Pack (www.virtualbox.org)
- At least 165 GB free disk space
- Bridged network adapter
- SSH access
- Login: dev
- Password: dev

SoMLabs virtual machine



From host machine:

- ssh dev@<ip_address>
- password: dev

or

- ssh dev@somlabs.local
- password: dev

Building system and SDK

Building system and SDK

- Installing required packages:

sudo apt-get install repo gawk wget git diffstat unzip texinfo gcc-multilib build-essential chrpath socat libsdl1.2-dev xterm sed cvs subversion coreutils texi2html docbook-utils python-pysqlite2 help2man make gcc g++ desktop-file-utils libgl1-mesa-dev libglu1-mesa-dev mercurial autoconf automake groff curl lzop asciidoc u-boot-tools

Building system and SDK

- Downloading the sources

```
mkdir imx-yocto-bsp

cd imx-yocto-bsp

repo init -u https://source.codeaurora.org/external/imx/imx-manifest -b imx-linux-zeus -m imx-5.4.3-2.0.0.xml

repo sync

cd sources

git clone -b zeus https://github.com/SoMLabs/imx-meta-somlabs.git meta-somlabs
```

Building system and SDK

- Configuring the build (imx-setup-release.sh)

```
echo "BBLAYERS += \"\${BSPDIR}/sources/meta-somlabs\"" >>
$BUILD_DIR/conf/bblayers.conf

echo "LICENSE_FLAGS_WHITELIST = \"commercial\"" >>
$BUILD_DIR/conf/local.conf
```

Building system and SDK

- Selecting the machine:

visionsom-8mm-cb-std

visioncb-6ull-std-emmc-btwifi

visioncb-6ull-std-emmc

visioncb-6ull-std-sd-btwifi

visioncb-6ull-std-sd

DISTRO=fsl-imx-wayland MACHINE=<SELECTED_MACHINE> source imx-setup-release.sh -b <BUILD_DIRECTORY>

Building system and SDK

- Building the system:

  bitbake fsl-image-validation-imx

  image is located in tmp/deploy/images

- Building the SDK:

  bitbake fsl-image-validation-imx -c populate_sdk

  installer is located in tmp/deploy/sdk

Building system and SDK

- Detailed instruction is located in the meta-somlabs repository README file and on wiki website:

  github.com/SoMLabs/imx-meta-somlabs/blob/zeus/README.md

  wiki.somlabs.com/index.php/VisionSOM_imx-meta-somlabs

Yocto project structure

Yocto project structure

```
.
├── build-visionsom-8mm-cb-std
│   ├── bitbake-cookerdaemon.log
│   ├── cache
│   ├── conf
│   ├── sstate-cache
│   └── tmp
├── fsl-imx-wayland-glibc-x86_64-fsl-image-validation-imx-aarch64-toolchain-5.4-zeus.sh
├── imx-setup-release.sh -> sources/meta-imx/tools/imx-setup-release.sh
├── README -> sources/base/README
├── README-IMXBSP -> sources/meta-imx/README
├── setup-environment -> sources/base/setup-environment
└── sources
    ├── base
    ├── meta-browser
    ├── meta-freescale
    ├── meta-freescale-3rdparty
    ├── meta-freescale-distro
    ├── meta-imx
    ├── meta-openembedded
    ├── meta-qt5
    ├── meta-rust
    ├── meta-somlabs
    ├── meta-timesys
    └── poky
```

Yocto project structure

```
sources/meta-imx/meta-bsp/
├── classes
├── conf
├── recipes-bsp
├── recipes-connectivity
├── recipes-core
├── recipes-devtools
├── recipes-graphics
├── recipes-kernel
├── recipes-multimedia
├── recipes-security
├── recipes-support
└── recipes-utils
```

```
sources/meta-imx/meta-bsp/recipes-kernel/
├── cryptodev
│   ├── cryptodev-linux
│   └── cryptodev-linux_1.10.bbappend
├── kernel-modules
│   ├── kernel-module-imx-gpu-viv
│   ├── kernel-module-imx-gpu-viv_6.4.0.p2.2.bb
│   ├── kernel-module-mwifiex_git.inc
│   ├── kernel-module-pcie8997.bb
│   ├── kernel-module-qca6174_3.0.bb
│   ├── kernel-module-qca9377_3.1.bb
│   ├── kernel-module-qcacld_3.1.inc
│   └── kernel-module-qcacld-lea.inc
├── linux
│   ├── linux-imx_5.4.bb
│   └── linux-imx-headers_5.4.bb
├── linux-firmware
│   ├── files
│   ├── linux-firmware_%.bbappend
│   └── sof-imx_1.4.1-392.bb
└── linux-libc-headers
    ├── linux-libc-headers
    └── linux-libc-headers_5.4.bb
```

Yocto project structure

```
# Copyright (C) 2013-2016 Freescale Semiconductor
# Copyright 2017-2019 NXP
# Released under the MIT license (see COPYING.MIT for the terms)

SUMMARY = "Linux Kernel provided and supported by NXP"
DESCRIPTION = "Linux Kernel provided and supported by NXP with focus on \
i.MX Family Reference Boards. It includes support for many IPs such as GPU, VPU and IPU."

require recipes-kernel/linux/linux-imx.inc

LIC_FILES_CHKSUM = "file://COPYING;md5=bbea815ee2795b2f4230826c0c6b8814"

DEPENDS += "lzop-native bc-native"

KERNEL_BRANCH ?= "imx_5.4.3_2.0.0"
LOCALVERSION = "-2.0.0"
KERNEL_SRC ?= "git://source.codeaurora.org/external/imx/linux-imx.git;protocol=https"
SRC_URI = "${KERNEL_SRC};branch=${KERNEL_BRANCH}"

SRCREV = "fd263a3edd95dfe812397fabf1059b5f99bba2ab"

FILES_${KERNEL_PACKAGE_NAME}-base += "${nonarch_base_libdir}/modules/${KERNEL_VERSION}/modules.builtin.modinfo "

KERNEL_CONFIG_COMMAND = "oe_runmake_call -C ${S} CC="${KERNEL_CC}" O=${B} olddefconfig"

FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}:"
```

SoMLabs meta-layer and hardware support

SoMLabs meta-layer and hardware support

- Support for SoMLabs modules and carrier boards:
  - VisionSOM-6ULL
  - VisionSOM-8Mmini
- Sources for kernel and u-boot:
  - github.com/SoMLabs/somlabs-linux-imx
  - github.com/SoMLabs/somlabs-uboot-imx
- Demo applications

SoMLabs meta-layer and hardware support

```
sources/meta-somlabs/
├── conf
│   ├── layer.conf
│   └── machine
│       ├── include
│       ├── visioncb-6ull-std-emmc-btwifi.conf
│       ├── visioncb-6ull-std-emmc.conf
│       ├── visioncb-6ull-std-sd-btwifi.conf
│       ├── visioncb-6ull-std-sd.conf
│       └── visionsom-8mm-cb-std.conf
├── custom-licenses
│   ├── Cortina
│   ├── Freescale-Binary-EULA
│   ├── Freescale-EULA
│   ├── NXP-Binary-EULA
│   └── TestFloat
├── EULA
├── README.md
├── recipes-bsp
│   ├── imx-atf
│   │   ├── imx-atf
│   │   └── imx-atf_2.0.bbappend
│   ├── imx-mkimage
│   │   └── imx-boot_%.bbappend
│   └── u-boot
│       ├── u-boot-imx
│       └── u-boot-imx_2019.04.bbappend
```

```
├── recipes-fsl
│   └── images
│       └── fsl-image-validation-imx.bbappend
├── recipes-graphics
│   └── wayland
│       ├── weston-init
│       └── weston-init.bbappend
├── recipes-kernel
│   ├── linux
│   │   └── linux-imx_5.4.bbappend
│   └── linux-firmware
│       └── linux-firmware_%.bbappend
└── recipes-somlabs
    └── somlabs-demo
        ├── somlabs-demo
        └── somlabs-demo.bb
```

SoMLabs meta-layer and hardware support

```
dev@somlabs:~/imx-yocto-bsp$ cat sources/meta-somlabs/conf/machine/include/visionsom-8mm-cb.inc
# Provides the VisionSOM-8MM-CB common settings

MACHINEOVERRIDES =. "mx8:mx8m:mx8mm:"

require conf/machine/include/imx-base.inc
require conf/machine/include/tune-cortexa53.inc

MACHINE_FEATURES += " pci optee"

UBOOT_CONFIG ??= "sd"
UBOOT_CONFIG[sd] = "visionsom_8mm_defconfig,sdcard"
SPL_BINARY = "spl/u-boot-spl.bin"

IMX_KERNEL_CONFIG_AARCH64 = "visionsom_8mm_defconfig"

DDR_FIRMWARE_NAME = "lpddr4_pmu_train_1d_imem.bin lpddr4_pmu_train_1d_dmem.bin lpddr4_pmu_train_2d_imem.bin lpddr4_pmu_train_2d_dmem.bin"

UBOOT_DTB_NAME = "visionsom-8mm.dtb"

IMXBOOT_TARGETS = "flash_evk"

SERIAL_CONSOLES = "115200;ttymxc3"

IMAGE_BOOTLOADER = "imx-boot"

LOADADDR = ""
IMX_BOOT_SEEK = "33"

IMAGE_BOOT_FILES = " \
        ${KERNEL_IMAGETYPE} \
        ${@make_dtb_boot_files(d)} \
"
```

SoMLabs meta-layer and hardware support

```
dev@somlabs:~/imx-yocto-bsp$ cat sources/meta-somlabs/conf/machine/visionsom-8mm-cb-std.conf
#@TYPE: Machine
#@NAME: SoMLabs i.MX8MM VisionCB-8M-STD
#@SOC: i.MX8MM
#@DESCRIPTION: Machine configuration for VisionCB-8M-STD board with VisionSOM-8MM
#@MAINTAINER: Krzysztof Chojnowski <krzysztof.chojnowski@somlabs.com>

MACHINEOVERRIDES =. "visionsom-8mm-cb:"

include conf/machine/include/visionsom-8mm-cb.inc

KERNEL_DEVICETREE = "freescale/visionsom-8mm-cb-std.dtb"

MACHINE_FEATURES += "wifi bluetooth bcm43430"
```

SoMLabs meta-layer and hardware support

```
dev@somlabs:~/imx-yocto-bsp$ cat sources/meta-somlabs/recipes-bsp/u-boot/u-boot-imx_2019.04.bbappend
FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}:"

UBOOT_SRC = "git://github.com/SoMLabs/somlabs-uboot-imx.git;protocol=https"
SRCBRANCH = "somlabs-lf-5.4.y_v2019.04"
SRC_URI = "${UBOOT_SRC};branch=${SRCBRANCH} \
           file://splash.bmp \
          "
SRCREV = "3d0c1acbcba88fd79c3d5c854593401510bcdb82"


do_compile_prepend_visioncb-6ull-std() {
        sed -i "s/setfdtfile=setenv fdt_file somlabs-\${board}\${fdt_suffix}.dtb/setfdtfile=setenv fdt_file ${KERNEL_DEVICETREE}/g" ${S}/include/configs/visionsom_6ull.h
}

do_install_append_visioncb-6ull-std() {
        install -d ${DEPLOY_DIR_IMAGE}
        install -m 0644 ${WORKDIR}/splash.bmp ${DEPLOY_DIR_IMAGE}/splash.bmp
}
```

SoMLabs meta-layer and hardware support

```
dev@somlabs:~/imx-yocto-bsp$ cat sources/meta-somlabs/recipes-kernel/linux/linux-imx_5.4.bbappend
FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}:"

LOCALVERSION = "-lts-${KERNEL_BRANCH}"
KERNEL_SRC = "git://github.com/SoMLabs/somlabs-linux-imx.git;protocol=http"
SRC_URI = "${KERNEL_SRC};branch=${KERNEL_BRANCH} \
           "

KERNEL_BRANCH = "somlabs_imx_5.4.24_2.1.0"
SRCREV = "d0ce74ff14da6634880b70698f17dd6efeb2816e"

addtask copy_somlabs_defconfig after do_patch copy_defconfig before do_preconfigure
do_copy_somlabs_defconfig () {
}

do_copy_somlabs_defconfig_append_visionsom-8mm-cb () {
    cp ${S}/arch/arm64/configs/visionsom_8mm_defconfig ${B}/.config
    cp ${S}/arch/arm64/configs/visionsom_8mm_defconfig ${B}/../defconfig
}

do_copy_somlabs_defconfig_append_visioncb-6ull-std () {
    cp ${S}/arch/arm/configs/visionsom_6ull_defconfig ${B}/.config
    cp ${S}/arch/arm/configs/visionsom_6ull_defconfig ${B}/../defconfig
}
```

System installation

System installation

- Copy image to host machine:

  scp dev@somlabs.local:imx-yocto-bsp/build-visionsom-8mm-cb-std/tmp/deploy/images/visionsom-8mm-cb-std/fsl-image-validation-imx-visionsom-8mm-cb-std.sdcard.bz2 .

- Extracting image file:

  bunzip2 -dkf fsl-image-validation-imx-visionsom-8mm-cb-std.sdcard.bz2

System installation

- Installing on SD-card:

```
sudo dd if=fsl-image-validation-imx-visionsom-8mm-cb-std.sdcard
of=/dev/sdX bs=1M status=progress
sudo sync
```

System installation

- Obtaining UUU tool from NXP:

  github.com/NXPmicro/mfgtools

  github.com/NXPmicro/mfgtools/releases

- Copying bootloader image to the host system:

  scp dev@somlabs.local:imx-yocto-bsp/build-visionsom-8mm-cb-std/tmp/deploy/images/visionsom-8mm-cb-std/imx-boot-visionsom-8mm-cb-std-sd.bin-flash_evk .

System installation

- Installing on eMMC:

  sudo ./uuu -v -b emmc_all imx-boot-visionsom-8mm-cb-std-sd.bin-flash_evk fsl-image-validation-imx-visionsom-8mm-cb-std.sdcard

```
1:82>Okay (0.103s)
1:82>Start Cmd:FB: ucmd setenv mmcdev ${emmc_dev}
1:82>Okay (0.004s)
1:82>Start Cmd:FB: ucmd mmc dev ${emmc_dev}
1:82>Okay (0.097s)
1:82>Start Cmd:FB: flash -raw2sparse all fsl-image-validation-imx-visionsom-8mm-cb-std.sdcard
100%1:82>Okay (150.8s)
1:82>Start Cmd:FB: flash bootloader imx-boot-visionsom-8mm-cb-std-sd.bin-flash_evk
0x400000001:82>Okay (0.262s)
1:82>Start Cmd:FB: ucmd if env exists emmc_ack; then ; else setenv emmc_ack 0; fi;
1:82>Okay (0.004s)
1:82>Start Cmd:FB: ucmd mmc partconf ${emmc_dev} ${emmc_ack} 1 0
1:82>Okay (0.007s)
1:82>Start Cmd:FB: done
1:82>Okay (0s)
```

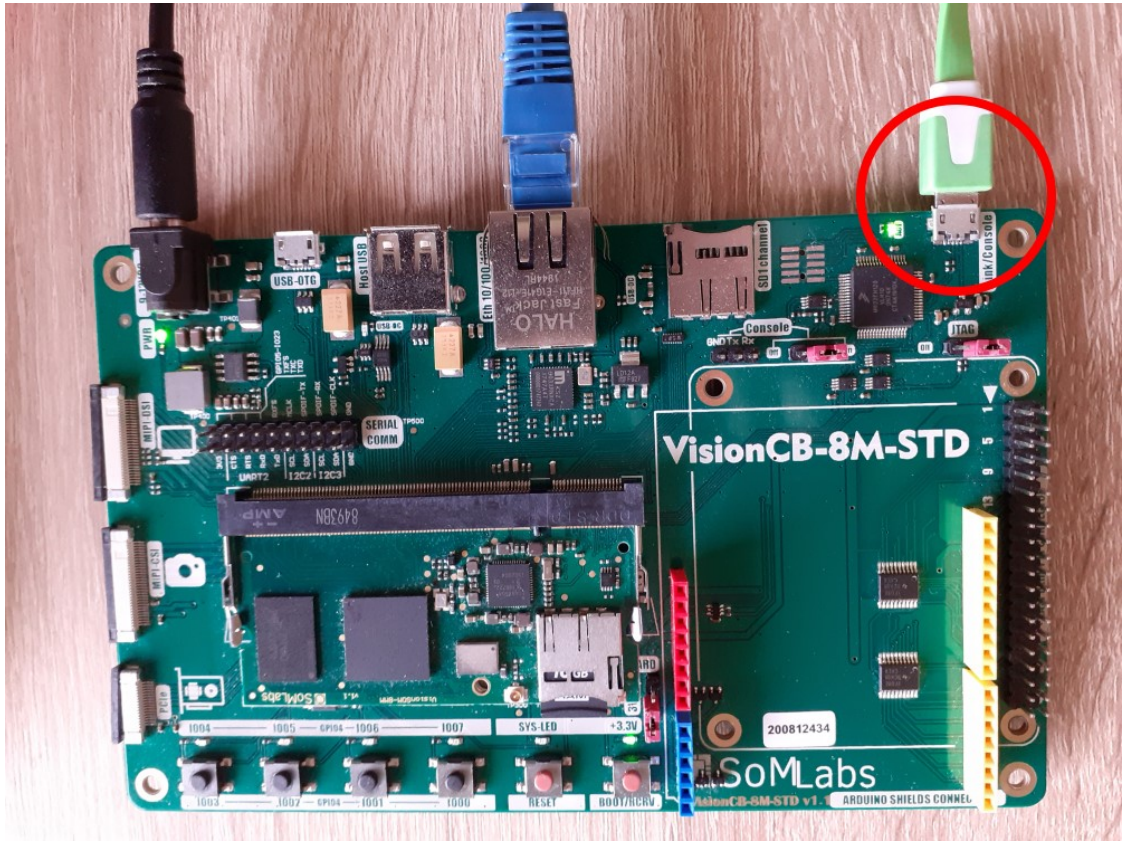
System installation

- Connecting to the system (serial port)

  sudo screen /dev/ttyACM0 115200

```
NXP i.MX Release Distro 5.4-zeus visionsom-8mm-cb-std ttymxc3

visionsom-8mm-cb-std login: root
Last login: Fri Sep 18 11:15:21 UTC 2020 on tty7
root@visionsom-8mm-cb-std:~#
```

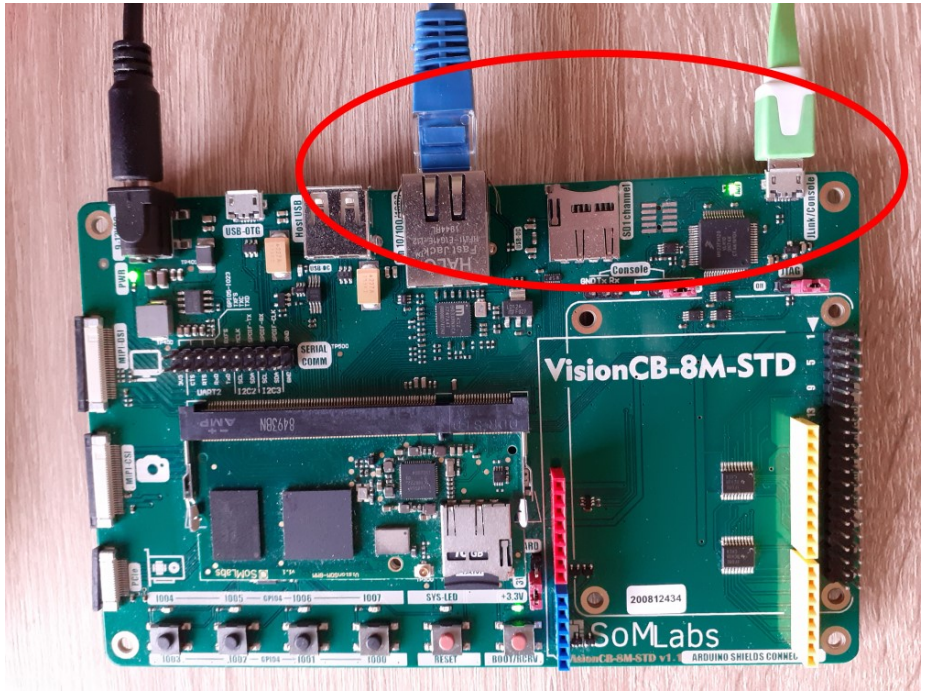


System installation

- Connecting to the system (SSH)
  - ssh root@<IP>
  - ssh root@visionsom-8mm-cb-std.local

```
root@visionsom-8mm-cb-std:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether c6:63:0b:78:c3:bb brd ff:ff:ff:ff:ff:ff
    inet 10.71.163.161/20 brd 10.71.175.255 scope global dynamic eth0
       valid_lft 172678sec preferred_lft 172678sec
    inet6 fe80::c463:bff:fe78:c3bb/64 scope link
       valid_lft forever preferred_lft forever
3: wlan0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 58:d5:0a:01:3e:b8 brd ff:ff:ff:ff:ff:ff
```

```
dev@somlabs:~$ ssh root@10.71.163.161
The authenticity of host '10.71.163.161 (10.71.163.161)' can't be established.
RSA key fingerprint is SHA256:9FB4D2lN/gOc+IywWz/716w3uPj6UkFMc017BGfIUm0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.71.163.161' (RSA) to the list of known hosts.
root@visionsom-8mm-cb-std:~#
```



Building C application with Yocto SDK

Building C application

- Configuring the environment

  . /opt/fsl-imx-wayland/5.4-zeus/environment-setup-aarch64-poky-linux

```
dev@somlabs:~/Excercises/Workshop1/Lab1$ echo $CC
aarch64-poky-linux-gcc -mcpu=cortex-a53+crc+crypto --sysroot=/opt/fsl-imx-wayland/5.4-zeus/sysroots/aarch64-poky-linux
```

Building C application

- Source code

  /home/dev/Excercises/Workshop1/Lab1/main.c

- Compilation

  $CC main.c -o hello

Building C application

- Running the example

```
dev@somlabs:~/Excercises/Workshop1/Lab1$ scp hello root@10.71.163.161:
hello                                         100%   13KB   2.2MB/s   00:00
dev@somlabs:~/Excercises/Workshop1/Lab1$ ssh root@10.71.163.161
root@visionsom-8mm-cb-std:~# ls
hello
root@visionsom-8mm-cb-std:~# ./hello
Hello VisionSOM!
```

Creating a new recipe

Creating a new recipe

- meta-somlabs/recipes-somlabs/somlabs-example:

```
somlabs-example/
├── somlabs-example
│   └── main.c
└── somlabs-example.bb
```

- /home/dev/Excercises/Workshop1/Lab2

```
cd ~/imx-yocto-bsp/sources/meta-somlabs/recipes-somlabs/
cp -r ~/Excercises/Workshop1/Lab2/somlabs-example .
```

Creating a new recipe

- /meta-somlabs/recipes-somlabs/somlabs-example/somlabs-example/main.c

```c
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>

#define LED_PATH "/sys/class/leds/LED-IO-05/brightness"
#define BLINKS_COUNT 3

int main(void) {

  int fd;
  fd = open(LED_PATH, O_WRONLY);
  if(fd < 0) {
    printf("Could not open the file\n");
    return -1;
  }

  printf("Blinking start\n");

  for(int i = 0; i< BLINKS_COUNT; i++) {
    write(fd, "1", 1);
    sleep(1);
    write(fd, "0", 1);
    sleep(1);
  }

  printf("Blinking stop\n");

  close(fd);

  return 0;
}
```

Creating a new recipe

- /meta-somlabs/recipes-somlabs/somlabs-example/somlabs-example.bb

```
DESCRIPTION = "Example application"
LICENSE = "BSD-3-Clause"
LIC_FILES_CHKSUM = "file://${COREBASE}/meta/files/common-licenses/BSD-3-Clause;md5=550794465ba0ec5312d6919e203a55f9"

inherit pkgconfig

SRC_URI = " \
    file://main.c \
    "

S = "${WORKDIR}"

do_compile() {
    ${CC} ${CFLAGS} ${LDFLAGS} main.c -o somlabs-example
}

do_install() {
    install -d ${D}/usr/share/somlabs-example/
    install -m 0755 somlabs-example ${D}/usr/share/somlabs-example/
}

FILES_${PN} = " /usr/share/somlabs-example/ "
```

Creating a new recipe

- meta-somlabs/recipes-fsl/images/fsl-image-validation-imx.bbappend

```
PACKAGE_INSTALL += " \
    somlabs-demo \
    somlabs-example \
    "
```

Creating a new recipe

- Building new image

```
ssh dev@somlabs.local

cd ~/imx-yocto-bsp

DISTRO=fsl-imx-wayland MACHINE=visionsom-8mm-cb-std source imx-setup-release.sh -b build-visionsom-8mm-cb-std

bitbake fsl-image-validation-imx
```

Creating a new recipe

- Installing new image

  scp dev@somlabs.local:imx-yocto-bsp/build-visionsom-8mm-cb-std/tmp/deploy/images/visionsom-8mm-cb-std/fsl-image-validation-imx-visionsom-8mm-cb-std.sdcard.bz2 .

  bunzip2 -dkf fsl-image-validation-imx-visionsom-8mm-cb-std.sdcard.bz2

  scp dev@somlabs.local:imx-yocto-bsp/build-visionsom-8mm-cb-std/tmp/deploy/images/visionsom-8mm-cb-std/imx-boot-visionsom-8mm-cb-std-sd.bin-flash_evk .

  sudo ./uuu -v -b emmc_all imx-boot-visionsom-8mm-cb-std-sd.bin-flash_evk fsl-image-validation-imx-visionsom-8mm-cb-std.sdcard

Autostart application

Creating a new recipe

- meta-somlabs/recipes-somlabs/somlabs-example:

```
somlabs-example/
├── somlabs-example
│   ├── main.c
│   └── somlabs-example.service
└── somlabs-example.bb
```

- /home/dev/Excercises/Workshop1/Lab3

```
cd ~/imx-yocto-bsp/sources/meta-somlabs/recipes-somlabs/
cp -r ~/Excercises/Workshop1/Lab3/somlabs-example .
```

Creating a new recipe

- meta-somlabs/recipes-somlabs/somlabs-example/somlabs-example.bb

```
DESCRIPTION = "Example application"
LICENSE = "BSD-3-Clause"
LIC_FILES_CHKSUM = "file://${COREBASE}/meta/files/common-licenses/BSD-3-Clause;md5=550794465ba0ec5312d6919e203a55f9"

inherit pkgconfig
inherit systemd

SYSTEMD_AUTO_ENABLE = "enable"
SYSTEMD_SERVICE_${PN} = "somlabs-example.service"

SRC_URI = " \
    file://main.c \
    file://somlabs-example.service \
    "

S = "${WORKDIR}"

do_compile() {
    ${CC} ${CFLAGS} ${LDFLAGS} main.c -o somlabs-example
}

do_install() {
    install -d ${D}/usr/share/somlabs-example/
    install -m 0755 somlabs-example ${D}/usr/share/somlabs-example/

    install -d ${D}/${systemd_unitdir}/system
    install -m 0644 ${WORKDIR}/somlabs-example.service ${D}/${systemd_unitdir}/system
}

FILES_${PN} = " /usr/share/somlabs-example/ ${systemd_unitdir}/system/somlabs-example.service"
```

Creating a new recipe

- meta-somlabs/recipes-somlabs/somlabs-example/somlabs-example/somlabs-example.service

```
[Unit]
Description=SoMLabs example application startup script

[Service]
ExecStart=/usr/share/somlabs-example/somlabs-example

[Install]
WantedBy=multi-user.target
```

- Connecting the Future
- Customer Centric