# introduction to the board and workspace setup

## goal

The goal of this Lab is to introduce you to the Raspberry Pi 3 board and to provide instructions about how to set up the workspace used for the following labs.

## pre-requisites

To follow this Lab, you will need:

1. Raspberry Pi 3 board full;
2. Micro USB cable;
3. 8 GB Micro SD card;
4. USB-to-Serial debug module for Raspberry Pi 3 or USB to TTL adapter;
5. Ethernet cable;
6. A PC with Ubuntu Desktop 14.04 LTS, or a virtual machine hosting Ubuntu Desktop 14.04 LTS with at least 100 GB of free space;
7. A Micro SD card reader attached to the PC/virtual machine;
8. (**Optional**) Micro HDMI cable.

## The Raspberry Pi 3

The Raspberry Pi 3 is the successor to the Raspberry Pi 2. It builds upon the Pi 2 by upgrading the Arm cores to Cortex-A53 and adding an onboard single-band 2.4GHz-only wireless chipset.

The Raspberry Pi 3 measures the same 85.60mm x 53.98mm x 17mm, with a little overlap for the SD card and connectors that project over the edges. The SoC is a Broadcom BCM2837. This contains a quad-core Cortex-A53 running at 1.2GHz and a Videocore 4 GPU.

|  |  |
| --- | --- |
|  |  |
| **Front View** |

## workplace setup

The setup is divided in two parts: PC setup and board setup.

### PC setup

The following operations must be done once, to set up the Linux distribution you are using to be ready for building embedded Linux on the Raspberry Pi 3.

Install the following packages:

sudo apt-get install gawk wget git diffstat unzip texinfo gcc-multilib build-essential

sudo apt-get install chrpath socat libsdl1.2-dev xterm ncurses-dev lzop

sudo apt-get install minicom u-boot-tools curl

You will need to install the latest version of python. Type python -–version to check, or use the following command to download it.

sudo apt-get install python

Install the ***repo*** utility:

mkdir ~/bin

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

chmod a+x ~/bin/repo

export PATH=$PATH:~/bin

If you are using Git for the first time in your system, you have to set your username and email address as follows:

git config --global user.name "username"

git config --global user.email “email@example.com”

Download the source code of embedded Linux for Raspberry Pi 3:

mkdir -p ~/raspberryPi3/sources

cd ~/raspberryPi3/sources

git clone -b krogoth git://git.yoctoproject.org/poky

git clone -b krogoth git://git.openembedded.org/meta-openembedded

git clone -b krogoth git://git.yoctoproject.org/meta-raspberrypi

**NOTE**: Be patient, this operation may take several minutes as several megabytes of source code have to be downloaded.

It is now time to define a specific network configuration for our lab experiments. For this purpose, let us create the following directory:

mkdir ~/raspberryPi3/sources/poky/meta/recipes-core/init-ifupdown/init-ifupdown-1.0/raspberrypi3

In this directory, place a file named “***interfaces***” with the following content:

# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)

# The loopback interface

auto lo

iface lo inet loopback

# Wireless interfaces

iface wlan0 inet dhcp

 wireless\_mode managed

 wireless\_essid any

 wpa-driver wext

 wpa-conf /etc/wpa\_supplicant.conf

iface atml0 inet dhcp

# Wired or wireless interfaces

auto eth0

iface eth0 inet static

 address 192.168.1.2

 netmask 255.255.255.0

 gateway 192.168.1.1

 network 192.168.1.0

# Ethernet/RNDIS gadget (g\_ether)

# ... or on host side, usbnet and random hwaddr

iface usb0 inet static

 address 192.168.7.2

 netmask 255.255.255.0

 network 192.168.7.0

 gateway 192.168.7.1

# Bluetooth networking

iface bnep0 inet dhcp

Among the other activities, with this file, we set the IP address for the Ethernet device to 192.168.1.2.

Configure the environment to build the source just downloaded for the Raspberry Pi 3 and accept the license agreement:

cd ~/raspberryPi3/

source sources/poky/oe-init-build-env rpi-build

Using your preferred editor program, edit the file “***rpi-build/conf/local.conf***”, remove the existing content, and type/paste in the following content, to include the needed information to build embedded Linux for the Raspberry Pi 3:

MACHINE ?= "raspberrypi3"

PREFERRED\_VERSION\_linux-raspberrypi = "4.%"

DISTRO\_FEATURES\_remove = "x11 wayland"

DISTRO\_FEATURES\_append = " systemd"

VIRTUAL-RUNTIME\_init\_manager = "systemd"

ENABLE\_UART = "1"

Using your favorite editor program, edit the file “***rpi-build/conf/bblayers.conf***”, remove the existing content, and type/paste in the following content, to include the needed information to build embedded Linux for the Raspberry Pi 3 (ensure it is formatted exactly as shown, spaces and missing indents can cause parse errors):

LCONF\_VERSION = "6"

BBPATH = "${TOPDIR}"

BSPDIR := "${@os.path.abspath(os.path.dirname(d.getVar('FILE', True)) + '/../..')}"

BBFILES ?= ""

BBLAYERS ?= " \

 ${BSPDIR}/sources/poky/meta \

 ${BSPDIR}/sources/poky/meta-yocto \

 ${BSPDIR}/sources/poky/meta-yocto-bsp \

 ${BSPDIR}/sources/meta-openembedded/meta-oe \

 ${BSPDIR}/sources/meta-openembedded/meta-multimedia \

 ${BSPDIR}/sources/meta-raspberrypi \

 "

Once this operation is completed, the environment is ready for building embedded Linux for the Raspberry Pi 3.

Ensure you are still in the build environment “rpi-build”. The Linux for the Raspberry Pi 3 can now be built by issuing the following command:

bitbake rpi-basic-image

**NOTE**: Be patient, this operation may take up to two hours, depending on the PC you are using, as several megabytes of source code have to be compiled. If you are compiling the image on a virtual machine, this may take several hours.

Once the build is completed, an SD card image for booting up the Raspberry Pi 3 can be found in the following path:

raspberryPi3/rpi-build/tmp-glibc/deploy/images/raspberrypi3/rpi-basic-image-raspberrypi3.rpi-sdimg

The image can be copied to a Micro SD using some Linux functionality (assuming the Micro SD is available to the PC as /dev/sdN). Alternatively, and more easily, a flash program could be used.

First, run the

sudo fdisk -l

command to determine which device to flash to (plug in and unplug the SD card to determine which device it is). For this example, the SD card is under the name “sdc” (this may be different in your environment). Next, ensure that the device is unmounted. This can be done using the command:

sudo umount /dev/sdc\*

Once this is done, the following command can be used to copy the image across to the SD card (substitute any folder names and device names to ensure they are relevant to your specific environment).

sudo dd bs=1M if=/home/user/raspberryPi3/rpi-build/tmp-glibc/deploy/images/raspberrypi3/rpi-basic-image-raspberrypi3.rpi-sdimg of=/dev/sdc

Note that if not done properly, the image being flashed across to the SD card may cause problems when attempting to turn on the board. If this is the case, it may be worth retrying the process again and ensuring that it is done properly, or use a flash program to automate the process.

While the SD card is still connected to the development host, use the following lines to navigate to the etc folder on the SD card (assuming the device has now been mounted).

cd /media/user/***SD\_name***/etc – Use the ls command to find the name of the SD card in the user folder.

Then, use a terminal text editor to open the shadow file.

sudo vi shadow

Or

sudo gedit shadow

Check that there are no characters between the first two colons in the first line. If there is, remove it so that the first line looks like this:

Root::17728:0:99999:7:::

Exit the text editor by entering :x or simply closing the application!

### Minicom setup

Connect the USB-to-Serial debug module for Raspberry Pi 3 or USB to TTL adapter to J8 on the Raspberry Pi 3 board (pins 8 and 10 are UART0\_TX and UART0\_RX, which are the lines where the serial console of BCM2837 is routed). And configure the communication parameter to 1152008N1 in your preferred serial terminal emulation program (under Linux uses the device /dev/ttyUSB0 or /dev/ttyUSB1).

Once the programming of the Micro SD is completed, we are ready to configure the communication program that we will use to connect to the board: minicom. Issue the following command to start minicom in setup mode:

sudo minicom -s

Then, perform the following steps:

1. Select “***Serial port setup***” in the configuration menu;
2. Press A to modify the Serial Device and to set it to “***/dev/ttyUSB0***”;
3. Press F to set Hardware Flow Control to “***No***”;
4. Press enter to return to the previous menu;
5. Select “***Save setup as dfl***”, to save the setup as default configuration;
6. Select “***Exit***” from Minicom.

**Optional**: Raspberry Pi 3 Ethernet interface is configured to acquire the IP address via DHCP. In case you intend to connect the board with your PC directly, you need to configure the Ethernet connection properly. It is advisable to create a new Ethernet connection on your PC using the following statically assigned IPv4 configuration:

IP Address: 192.168.1.1
Netmask: 255.255.255.0
Gateway: 192.168.1.1

### Board setup

1. Insert the newly written Micro SD into the SD slot of the Raspberry Pi 3;
2. (**Optional**) connect the Micro HDMI cable to the Raspberry Pi 3 connector;
3. Connect a Micro USB cable to the PC and to the Micro USB connector of the Raspberry Pi 3;
4. Also connect the board with your host device via an ethernet cable.

Once the Micro USB cable is connected, the board is powered up. In case you connected the Micro HDMI cable, you will see a splash screen on the monitor, and a progression bar indicating the bootstrap is running.

Once the bootstrap is completed, a new device will be connected to the PC: /dev/ttyUSB0. The device is the virtual serial port through which you can access the Raspberry Pi 3, by issuing the following command:

sudo minicom

In case all previous steps have been completed successfully, you will see the following messages appearing on the terminal:

Welcome to minicom 2.7

OPTIONS: I18n

Compiled on Jan 1 2014, 17:13:19.

Port /dev/ttyUSB0, 10:22:24

Press CTRL-A Z for help on special keys

[ 0.000000] Booting Linux on physical CPU 0x0

[ 0.000000] Initializing cgroup subsys cpuset

[ 0.000000] Initializing cgroup subsys cpu

[ 0.000000] Initializing cgroup subsys cpuacct

......

Poky (Yocto Project Reference Distro) 2.1.2 raspberrypi3 ttyS0

raspberrypi3 login:

You can now login as “***root***” (no password required).

Note:

* To open the Minicom menu, press simultaneously ctrl + A and then Z (either lowercase or uppercase).

**Optional**: Raspberry Pi 3 Ethernet interface is configured to acquire the IP address via DHCP. In case you intend to connect the board with your PC directly, you need to configure the Ethernet connection properly. It is advisable to create a new Ethernet connection on the Raspberry Pi 3 by editing the file ***/etc/network/interfaces*** and to modify the configuration of the network device eth0 as follows:

# Wired or wireless interfaces

auto eth0

iface eth0 inet static

 address 192.168.1.2

 netmask 255.255.255.0

 gateway 192.168.1.255

Test to see if the network is configured properly using:

ifconfig

You should see an address of 19.168.1.2; however, if this is not the case, you can run the commands:

ifdown eth0

ifup eth0

After running these two commands, it should be configured properly.

If you are using a Linux virtual machine as the host, for the sole-purpose of communicating between the board and the host, you may need to switch the “Network” settings to a “Bridged Adapter” and select the ethernet device being used.

Once the PC and the Raspberry Pi 3 are configured, you can test the connection with the following command:

ping 192.168.1.1

If the configuration is correct, you will see a message appearing on the terminal similar to the following:

root@raspberrypi3:~# ping 192.168.1.1

PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.

64 bytes from 192.168.1.1: icmp\_seq=1 ttl=64 time=1.54 ms

64 bytes from 192.168.1.1: icmp\_seq=2 ttl=64 time=1.61 ms

64 bytes from 192.168.1.1: icmp\_seq=3 ttl=64 time=1.98 ms

64 bytes from 192.168.1.1: icmp\_seq=4 ttl=64 time=1.73 ms

# Post-lab practice

In this session, we have built an initial version of Linux image with Yocto Project. One of the main benefits of using Yocto Project is that the Linux image can be flexibly customized to accommodate specific usage; the key to manipulate it is by including different layers and recipes.

There are two files that determine the above feature, conf/local.conf and conf/bblayers.conf; to have a better understanding of these two files and the customization feature of Yocto Project, we shall carry out additional practice.