# ROW HAMMER LAB EXPERIENCE

This material was originally developed as part of an assignment of the Operating systems for embedded systems course delivered at Politecnico di Torino by Prof. Stefano Di Carlo.

Group Composed by:

- Niccolò Cacioli
- Michela Cavigliano
- Marco Rosa Gobbo
- Emanuele Bongiovanni

### 0) Prerequisites

- Around 80GB of free space for the virtual machine

-A raspberry pi4

-Micro SD card (at least 8gb)

### 1)First step: set up virtual machine for yocto

Install Virtual Box

Set up a new VM with a 80GB disk using ubuntu 22 LTS

Install the needed dependencies:

```
sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-
multilib build-essential chrpath socat cpio python3 python3-pip
python3-pexpect xz-utils debianutils iputils-ping python3-git python3-
jinja2 libegl1-mesa libsdl1.2-dev pylint3 xterm
```

Install the latest version of pylint:

```
sudo apt-get install pylint
which pylint
cd /usr/bin/
sudo ln -s pylint pylint3
```

Create the yocto directory and git clone the necessary files:

```
mkdir yocto
cd yocto
mkdir sources

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

git clone git://git.yoctoproject.org/meta-raspberrypi -b dunfell
```

```
git clone https://git.openembedded.org/meta-openembedded -b dunfell
```

## 2)Second step: configuring yocto and build image

Go back to the to project folder, initialize the build environment

```
cd ..
. sources/poky/oe-init-build-env
```

Edit the bblayers.conf in the conf folder with the layers that are needed for the raspberry.

```
BBLAYERS
?="\

              ${TOPDIR}/../sources/poky/meta \

              ${TOPDIR}/../sources/poky/meta-poky \

              ${TOPDIR}/../sources/poky/meta-yocto-bsp \

              ${TOPDIR}/../sources/meta-raspberrypi \

              ${TOPDIR}/../sources/meta-openembedded/meta-oe \

              ${TOPDIR}/../sources/meta-openembedded/meta-multimedia
            \

              ${TOPDIR}/../sources/meta-openembedded/meta-networking
            \

              ${TOPDIR}/../sources/meta-openembedded/meta-python \

              "
```

Edit the local.conf file the conf folder to set the target machine ( in our case raspberrypi4-64)

And add the extra OS features needed to work on it

```
MACHINE ?= "raspberrypi4-64"

…
```

```
EXTRA_IMAGE_FEATURES += "debug-tweaks tools-debug eclipse-debug ssh-
server-openssh"
```

Finally time to bake our image :

```
bitbake core-image-base
```

This may take more than an hour for the first build depending on the pc and the allocated resources to the VM

### 3)Third step: Flashing the sd card

From Ubuntu disk manager or from command line format the sd card

From the the image location run the bmap tool to flash the sd card

(you may need to change the destination if the usb key is not installed on /dev/sdb

```
Cd Desktop/yocto/build/tmp/deploy/images/raspberrypi4-64

sudo bmaptool copy --bmap core-image-full-cmdline-raspberrypi3-
64.wic.bmap    core-image-full-cmdline-raspberrypi3-64.wic.bz2 /dev/sdb
```

### 4)Fourth step: start programming!

You should now be able to boot into the minimal linux installation on the pi4, the account name is root and there is no password, using ifconfig you can setup the ethernet interface so to able to access the pi4 via ssh from another PC to work more easily

The basic instruction that you are going to need are the cache flush instructions, that you can find here : [https://developer.arm.com/documentation/den0024/a/Caches/Cache-maintenance](https://developer.arm.com/documentation/den0024/a/Caches/Cache-maintenance)

The pseudo code that needs to be implemented is this :

put addr1 into X9

for i := 0 to N − 1 do

LDR X0, [X9]

DC CIVAC, X9

DC CIVAC, X10

### 5)Optional: integration of the program in the build

In the image you will find gcc so you can work directly there, otherwise you can combine the binary file directly from yocto when building the image:

```
cd /yocto/
. sources/poky/oe-init-build-env

bitbake-layers create-layer ../meta-example

bitbake-layers add-layer ../meta-example/

cd ../meta-example/recipes-example mkdir myhello cd myhello
```

In the myhello folder you can place the recipe for the application, create a myhello.bb file to configure the recipe:

```
DESCRIPTION = "Simple helloworld application"

 LICENSE = "MIT"

LIC_FILES_CHKSUM =
"file://${COMMON_LICENSE_DIR}/MIT;md5=0835ade698e0bcf8506ecda2f7b4f302"
SRC_URI = "file://myhello.c"

S = "${WORKDIR}"

do_compile() {

 set CFLAGS -g ${CC} ${CFLAGS} myhello.c ${LDFLAGS} -o myhello unset
CFLAGS

}

 do_install() {

install -d ${D}${bindir} install -m 0755 myhello ${D}${bindir}

 }
```

```
mkdir files
```

We can now add the app to the conf/local.conf file :

```
IMAGE_INSTALL_append = "myhello"
```

Finally we can build again the image. This last part was taken from lab EL 01 of the operating systems course.